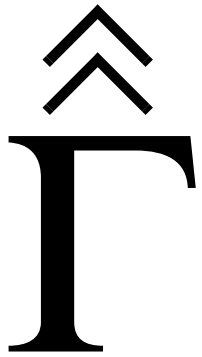
GAMMA

Common Relaxation Equations

Isotropic Liquid Systems



Author: Scott A. Smith

Date: June 8, 1998

Table of Contents

1 Introduction

This book is part of the GAMMA NMR simulation platform[1]. It deals exclusively with the theory and simulation of NMR relaxation and exchange. With GAMMA users may readily build programs to simulate a wide variety of relaxation phenomena. These can range in complexity from a relatively simple treatment using the “phenomenological” Bloch equations to a full blown Liouville space Redfield treatment using superoperators.

1.1 Relaxation Equation(s) Based on Simple Models

We can jump up one level in our treatment of relaxation by using a quantum mechanical approach but use simple models for what the spins do. For example, using motional model of a spherical top diffusing in an isotropic liquid we can exactly treat a case of dipolar relaxation effects. This allows one to make explicit formulae for expected T1, T2, and NOE values. The same can be said for other relaxation mechanisms (based on single spins or isolated spin pairs)and we can use this “single spin” and two spin” approach to deduce what will occur in multi-spin systems. This will of course not account for cross-correlation effects nor the effects of asymmetric motions.

What GAMMA provides is quite simple: The user works with a spin system in his/her program. The system contains the information (inter-nuclear distances, CSA values, quadrupolar couplings, a correlation time.....) necessary to perform computations of T1, T2, and NOE values using simple models. Several GAMMA functions are provided which take a spin system as a function argument, compute the desired value(s), and return those values to the user.

1. GAMMA is computational platform designed for the simulation of NMR phenomena by Smith, Levante, Meier and Ernst.

2 Common Dipolar Relaxation Equations

Relaxation by dipolar interactions is the most commonly treated relaxation mechanism in NMR. This chapter discusses the GAMMA module that supplies commonly used dipolar relaxation equations. In most cases the equations were derived using a quantum mechanical treatment on a single spin pair that is dynamically moving as a randomly diffusing spherical top. In multiple spin systems the relaxation values returned by these functions employ a sum over s spin pairs.

2.1 Available Dipolar Relaxation Functions

2.2 Covered Dipolar Relaxation Theory

2.3 Dipolar Relaxation Figures

2.4 Dipolar Relaxation Example Programs

2.5 Dipolar Relaxation Equations

2.5.1 R1_DD

Usage:

```
#include <gamma.h>
row_vector R1_DD(sys_dynamic &dsys);
double R1_DD(sys_dynamic &dsys, int spin1);
double R1_DD(sys_dynamic &dsys, int spin1, int spin2);
```

Description:

The function ***R1_DD*** returns a value or values for the longitudinal relaxation rate(s) expected from dipole-dipole interactions for the spin in the system ***dsys***. Returned units will be inverted seconds.

1. R1_DD(sys_dynamic &dsys) - The longitudinal relaxation rates of all spins in the system are returned in a row vector. Each spin is assumed interacting with all other spins and a two-spin approximation is used.
2. double R1_DD(spin_sys &sys, int spin) - The longitudinal relaxation rate resulting from dipole-dipole interactions for spin ***spin*** is returned based on a two-spin approximation.
3. double R1_DD(sys_dynamic& dsys, int spin1, int spin2) - The longitudinal relaxation rate resulting from the dipole-dipole interaction between ***spin1*** and ***spin2*** is returned.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value stored in ***dsys***, i.e. its τ_x value.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <gamma.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
row_vector R1s = R1_DD(sys);               // Vector of relaxation rates
double R10 = R1_DD(sys, 0);                // Relaxation rate of spin 1
double R101 = R1_DD(sys,0,1);              // Relaxation rate of spin 1 by spin 2
```

Mathematical Basis:

For an isotropic spherical top, the transverse relaxation rate expected from the dipole-dipole interaction of two spins will depend upon whether the two spins are like (*DDL*), or unlike (*DDU*). The corresponding formulae are given below for a spin which is being relaxed by another spin of spin quantum number S.

$$R_1^{DDU} = \frac{1}{T_1^{DDU}} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[\frac{2}{1+\Delta\omega_{IS}^2\tau^2} + \frac{6}{1+(\omega_I\tau)^2} + \frac{12}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_1^{DDL} = \frac{1}{T_1^{DDU}} = \frac{\xi^2\tau}{12\pi}I(I+1)\left[\frac{1}{1+(\omega_I\tau)^2} + \frac{4}{1+(2\omega_I\tau)^2}\right]$$

The dipolar interaction constant, ξ_{ij}^{D}, as defined in GAMMA is given by Eq. (1-1)(1-1) on page 18. Application of the two-spin approximation produces the formula utilized when there are multiple spin pairs in the system.

$$R_1^{DD}(i) = \sum_{i>i}^{spins} R_1^{DD}(i,j)$$

2.5.2 R1_DD_max

Usage:

```
#include <gamma.h>
double R1_DD_max(sys_dynamic &dsys);
```

Description:

The function ***R1_DD_max*** returns a value for the maximum longitudinal relaxation rate expected from dipole-dipole interactions. The function compares the rates for each spin in the input system ***dsys***.

Return Value:

A double precision number is returned.

Example:

```
#include <gamma.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
double Rmax = R1_DD_max(sys);              // Maximum relaxation rate
```

Mathematical Basis:

See the function description for ***R1_DD***.

2.5.3 R2_DD

Usage:

```
#include <gamma.h>
row_vector R2_DD(sys_dynamic &dsys);
double R2_DD(sys_dynamic &dsys, int spin1);
double R2_DD(sys_dynamic &dsys, int spin1, int spin2);
```

Description:

The function ***R2_DD*** returns a value(s) for the transverse relaxation rate expected from dipole-dipole interactions.

1. R2_DD(sys_dynamic &dsys) - The transverse relaxation rates of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double R2_DD(spin_sys &sys, int spin) - The transverse relaxation rate resulting from dipole-dipole in-

teractions for spin ***spin*** of system ***dsys*** is returned based on a two-spin approximation.

3. double R2_DD(sys_dynamic& dsys, int spin1, int spin2) - The transverse relaxation rate resulting from the dipole-dipole interaction between ***spin1*** and ***spin2*** of system ***dsys*** is returned.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <gamma.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
row_vector R2s = R2_DD(sys);               // Vector of relaxation rates
double R2i0 = R2_DD(sys, 0);               // Relaxation rate of spin 1
double R201 = R2_DD(sys,0,1);              // Relaxation rate of spin 1 by spin 2
```

Mathematical Basis:

For an isotropic spherical top, the transverse relaxation rate expected from the dipole-dipole interaction of two spins will depend upon whether the two spins are like (*DDL*), unlike (*DDU*), or unlike with resolved scalar coupling (*DDJ*). The corresponding formulae are given below for a spin which is being relaxed by another spin of spin quantum number s.

$$R_2^{DDU} = \frac{1}{T_2^{DDU}} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[4 + \frac{1}{1+\Delta\omega_{IS}^2\tau^2} + \frac{3}{1+\omega_I^2\tau^2} + \frac{6}{1+\omega_S^2\tau^2} + \frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_2^{DDJ} = \frac{1}{T_2^{DDJ}} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[4 + \frac{1}{1+\Delta\omega_{IS}^2\tau^2} + \frac{3}{1+\omega_I^2\tau^2} + \frac{3}{1+\omega_S^2\tau^2} + \frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_2^{DDL} = \frac{1}{T_2^{DDL}} = \frac{\xi^2\tau}{24\pi}S(S+1)\left[3 + \frac{5}{1+(\omega_I\tau)^2} + \frac{2}{1+(2\omega_I\tau)^2}\right]$$

The dipolar interaction constant, ξ_{ij}^D, as defined in GAMMA is given by Eq. (1-1) on page 18. Application of the two-spin approximation produces the formula utilized when there are multiple spin pairs in the system.

$$R_2^{DD}(i) = \sum_{i>i}^{spins} R_2^{DD}(i,j)$$

Two spins are considered "like" if the are the same isotope types, regardless of any chemical shift differences. Thus, spins are unlike if they are different isotope types. Scalar coupling is considered only in the case of two unlike spins.

2.5.4 R2_DD_max

Usage:

```
#include <gamma.h>
double R2_DD_max(sys_dynamic &dsys);
```

Description:

The function ***R2_DD_max*** returns a value for the maximum transverse relaxation rate expected from dipole-dipole interactions. The function compares the rates for each spin in the input system ***dsys***.

Return Value:

A double precision number is returned.

Example:

```
#include <gamma.h>
sys_dynamic dsys;                    // Set up a dynamic system
dsys.read("filename.sys");           // Read in system from file
double Rmax = R2_DD_max(sys);        // Maximum relaxation rate
```

Mathematical Basis:

See the function description for ***R2_DD***.

2.5.5 T1_DD

Usage:

```
#include <gamma.h>
row_vector T1_DD(sys_dynamic &dsys);
double T1_DD(sys_dynamic &dsys, int spin1);
double T1_DD(sys_dynamic &dsys, int spin1, int spin2);
```

Description:

The function ***T1_DD*** returns a value(s) for the longitudinal relaxation time expected from dipole-dipole interactions.

1. T1_DD(sys_dynamic &dsys) - The longitudinal relaxation times of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double T1_DD(spin_sys &sys, int spin) - The longitudinal relaxation time resulting from dipole-dipole interactions for spin ***spin*** of system ***dsys*** is returned based on a two-spin approximation.
3. double T1_DD(sys_dynamic& dsys, int spin1, int spin2) - The longitudinal relaxation time resulting from the dipole-dipole interaction between ***spin1*** and ***spin2*** of system ***dsys*** is returned.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <gamma.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
row_vector T1s = T1_DD(sys);               // Vector of relaxation times
double T10 = T1_DD(sys, 0);                // Relaxation time of spin 1
double T1i01 = T1_DD(sys,0,1);             // Relaxation time of spin 1 by spin 2
```

Mathematical Basis:

The longitudinal relaxation time is the inverse longitudinal relaxation rate. The two applicable equations are shown below for a single spin pair (left) and multiple spin pairs involving spin i (right).

$$T_1 = \frac{1}{R_1} \qquad 1/T_1^{DD}(i) = R_1^{DD}(i)$$

See the associated R_1 functions (***R1_DD***) for the relaxation rate formulae.

2.5.6 T1_DD_max

Usage:

```
#include <gamma.h>
double T1_DD_max(sys_dynamic &dsys);
```

Description:

The function ***T1_DD_max*** returns a value for the maximum longitudinal relaxation time expected from dipole-dipole interactions. The function compares the times for each spin in the input system ***dsys***.

Return Value:

A double precision number is returned.

Example:

```
#include <gamma.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
double Tmax = T1_DD_max(sys);              // Maximum relaxation time
```

Mathematical Basis:

See the function description for ***T1_DD***.

2.5.7 T2_DD

Usage:

```
#include <gamma.h>
row_vector T2_DD(sys_dynamic &dsys);
double T2_DD(sys_dynamic &dsys, int spin1);
double T2_DD(sys_dynamic &dsys, int spin1, int spin2);
```

Description:

The function ***T2_DD*** returns a value(s) for the transverse relaxation time expected from dipole-dipole interactions.

1. R2_DD(sys_dynamic &dsys) - The transverse relaxation times of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double R2_DD(spin_sys &sys, int spin) - The transverse relaxation time resulting from dipole-dipole interactions for spin ***spin*** of system ***dsys*** is returned based on a two-spin approximation.
3. double R2_DD(sys_dynamic& dsys, int spin1, int spin2) - The transverse relaxation time resulting from the dipole-dipole interaction between ***spin1*** and ***spin2*** of system ***dsys*** is returned.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <gamma.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
row_vector T2s = T2_DD(sys);               // Vector of relaxation times
double T20 = T2_DD(sys, 0);                // Relaxation time of spin 1
double T201 = T2_DD(sys,0,1);              // Relaxation time of spin 1 by spin 2
```

Mathematical Basis:

The transverse relaxation time is the inverse transverse relaxation rate. The two applicable equations are shown below for a single spin pair (left) and multiple spin pairs involving spin i (right).

$$T_2 = \frac{1}{R_2} \qquad 1/T_2^{DD}(i) = R_2^{DD}(i)$$

See the associated R_2 functions (***R2_DD***) for the relaxation rate formulae.

For an isotropic spherical top, the transverse relaxation time expected from the dipole-dipole interaction of two spins will depend upon whether the two spins are like (DDL), unlike (DDU), or unlike with resolved scalar coupling (DDJ). The corresponding formulae are given below for a spin which is being relaxed by another spin of spin quantum number s.

2.5.8 T2_DD_max

Usage:

```
#include <gamma.h>
double T2_DD_max(sys_dynamic &dsys);
```

Description:

The function ***T2_DD_max*** returns a value for the maximum transverse relaxation time expected from dipole-dipole interactions. The function compares the times for each spin in the input system ***dsys***.

Return Value:

A double precision number is returned.

Example:

```
#include <gamma.h>
sys_dynamic dsys;                    // Set up a dynamic system
dsys.read("filename.sys");           // Read in system from file
double Tmax = T2_DD_max(sys);        // Maximum relaxation time
```

Mathematical Basis:

See the function description for ***T2_DD***.

2.5.9 LWhh_DD

Usage:

```
#include <gamma.h>
row_vector LWhh_DD(sys_dynamic &dsys);
double LWhh_DD(sys_dynamic &dsys, int spin1);
double LWhh_DD(sys_dynamic &dsys, int spin1, int spin2);
```

Description:

The function ***LWhh_DD*** returns a value(s) for the linewidths (at half-height) expected from dipole-dipole interactions.

1. LWhh_DD(sys_dynamic &dsys) - The linewidths of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double LWhh_DD(spin_sys &sys, int spin) - The linewidth resulting from dipole-dipole interactions for spin ***spin*** of system ***dsys*** is returned based on a two-spin approximation.
3. double LWhh_DD(sys_dynamic& dsys, int spin1, int spin2) - The linewidth resulting from the dipole-dipole interaction between ***spin1*** and ***spin2*** of system ***dsys*** is returned.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <gamma.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
row_vector LWs = LWhh_DD(sys);             // Vector of Linewidths
double LW0 = LWhh_DD(sys, 0);              // Linewidth of spin 1
double LW01 = LWhh_DD(sys,0,1);            // Linewidth spin 1 due to spin 2
```

Mathematical Basis:

For an isotropic spherical top, the transverse relaxation rate expected from the dipole-dipole interaction of two spins will depend upon whether the two spins are like (*DDL*), unlike (*DDU*), or unlike with resolved scalar coupling (*DDJ*). The corresponding formulae are given below for a spin which is being relaxed by another spin of spin quantum number s.

$$R_2^{DDU} = \frac{1}{T_2^{DDU}} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[4 + \frac{1}{1+\Delta\omega_{IS}^2\tau^2} + \frac{3}{1+\omega_I^2\tau^2} + \frac{6}{1+\omega_S^2\tau^2} + \frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_2^{DDJ} = \frac{1}{T_2^{DDJ}} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[4 + \frac{1}{1+\Delta\omega_{IS}^2\tau^2} + \frac{3}{1+\omega_I^2\tau^2} + \frac{3}{1+\omega_S^2\tau^2} + \frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_2^{DDL} = \frac{1}{T_2^{DDL}} = \frac{\xi^2\tau}{24\pi}S(S+1)\left[3 + \frac{5}{1+(\omega_I\tau)^2} + \frac{2}{1+(2\omega_I\tau)^2}\right]$$

The dipolar interaction constant, ξ_{ij}^D, as defined in GAMMA is given by Eq. (1-1) on page 18 and application of the two-spin approximation produces

$$R_2^{DD}(i) = \sum_{i>i}^{spins} R_2^{DD}(i,j)$$

The line-width at half-height is related to the transverse relaxation rate by the simple formula

$$LW_{hh}^{DD} = R_2^{DD}/\pi$$

which under the two spin approximation is

$$LW_{hh}^{DD}(i) = R_2^{DD}(i)/\pi = \left(\sum_{j>i}^{spins} R_2^{DD}(i,j)\right)/\pi$$

2.5.10 LWhh_DD_max

Usage:

```
#include <gamma.h>
double LWhh_DD_max(sys_dynamic &dsys);
```

Description:

The function ***LWhh_DD_max*** returns a value for the maximum linewidth expected from dipole-dipole interactions. The function compares the times for each spin in the input system ***dsys***.

Return Value:

A double precision number is returned.

Example:

```
#include <gamma.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
double LWmax = LWhh_DD_max(sys);           // Maximum linewidth
```

Mathematical Basis:

See the function description for ***LWhh_DD***.

2.5.11 R2_DDMQT

Usage:

```
#include <gamma.h>
row_vector R2_DDMQT(sys_dynamic &dsys, int MQC);
double R2_DDMQT(sys_dynamic &dsys, int MQC, int spin1);
double R2_DDMQT(sys_dynamic &dsys, int MQC, int spin1, int spin2);
```

Description:

The function ***R2_DDMQT*** returns a value(s) for the transverse relaxation rate expected for multiple quantum transitions from dipole-dipole interactions.

1. R2_DDMQT(sys_dynamic &dsys, int MQC) - The transverse relaxation rates of the multiple quantum transition of order ***MQC*** for all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double R2_DDMQT(spin_sys &sys, int MQC, int spin) - The transverse relaxation rate of the multiple quantum transition of order ***MQC*** resulting from dipole-dipole interactions for spin ***spin*** of system ***dsys*** is returned based on a two-spin approximation.
3. double R2_DDMQT(sys_dynamic& dsys, int MQC, int spin1, int spin2) - The transverse relaxation rate of the multiple quantum transition of order ***MQC*** resulting from the dipole-dipole interaction between ***spin1*** and ***spin2*** of system ***dsys*** is returned.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <gamma.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
```

```
row_vector R = R2_DDMQT(sys,2);          // Vector of DQT relaxation rates
double R0 = R2_DDMQT(sys,0, 0);          // ZQT relax. rates involving spin 1
double R01 = R2_DDMQT(sys,0,0,1);        // ZQT relax. rate, spin 1 and spin2
```

Mathematical Basis:

For an isotropic spherical top, formulae for the transverse relaxation rates of the multiple quantum transitions of order MQC expected from the dipole-dipole interactions of two spins are shown below for a spin which is being relaxed by another spin of spin quantum number S.

$$[R_2^{DDJ}]^{ZQT} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[\frac{2}{1+[(\omega_I-\omega_S)\tau]^2}+\frac{3}{1+(\omega_I\tau)^2}+\frac{3}{1+(\omega_S\tau)^2}\right]$$

$$[R_2^{DDJ}]^{SQT} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[4+\frac{1}{1+\Delta\omega_{IS}^2\tau^2}+\frac{3}{1+(\omega_I\tau)^2}+\frac{3}{1+(\omega_S\tau)^2}+\frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$[R_2^{DDJ}]^{DQT} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[\frac{3}{1+(\omega_I\tau)^2}+\frac{3}{1+(\omega_S\tau)^2}+\frac{12}{1+[(\omega_I+\omega_S)\tau]^2}\right]$$

The superscript DDJ is left on as a reminder that these equations apply for spins with resolved scalar coupling. The dipolar interaction constant, ξ_{ij}^D, as defined in GAMMA is given by Eq. (1-1) on page 18 and application of the two-spin approximation produces

$$R_2^{DD}(i) = \sum_{j>i}^{spins} R_2^{DD}(i,j)$$

2.5.12 NOE

Usage:

```
#include <gamma.h>
double NOE(sys_dynamic &dsys, int spin1, int spin2, int eta=0);
```

Description:

The function ***NOE*** returns a value for the nuclear Overhauser enhancement to ***spin1*** resulting from the dipolar interaction with s***pin2***. The computation takes the spins from the system ***dsys***. It is assumed that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***. The internuclear separation between the spins is also obtained from ***dsys***. The value returned is given by

$$\rho_i = \frac{\gamma_j\left[\frac{6}{1+(\omega_i+\omega_j)^2\tau^2}-\frac{1}{1+\Delta\omega_{ij}^2\tau^2}\right]}{\gamma_i\left[\frac{1}{1+\Delta\omega_{ij}^2\tau^2}+\frac{3}{1+(\omega_i\tau)^2}+\frac{6}{1+(\omega_i+\omega_j)^2\tau^2}\right]}$$

unless a non-zero value of ***eta*** is supplied as an argument. For non-zero ***eta*** the value returned is given by

$$\eta_{NOE} = 1 + \rho$$

Return Value:

A double precision number is returned.

Examples:

```
#include <gamma.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
double rho = NOE(dsys, 0, 1);              // Get NOE of spin 0 due to spin 1
double eta = NOE(dsys, 0, 1, 1);           // Get eta NOE of spin 0 due to spin 1
```

2.6 Dipolar Relaxation Discussion

2.6.0.1 Dipolar Relaxation Sections

2.6.0.2 Dipolar Relaxation Figures

2.6.0.3 Dipolar Relaxation Example Programs

2.6.1 The Dipolar Interaction Constant

The dipolar interaction constant, ξ_{ij}^{D}, is used throughout GAMMA. It is simply a scaling factor which allows for independent scaling of spatial and spin tensors associated with the dipolar Hamiltonian (and others). Those interested in its origin must peruse the GAMMA documentation on the dipolar interaction. Since this constant is implicit, rather than explicit, to the functions described in this chapter, we merely present what it is.

$$\xi_{ij}^{D} = -2\sqrt{\frac{6\pi}{5}}\left(\frac{\mu_0}{4\pi}\right)h\frac{\gamma_i\gamma_j}{r_{ij}^3} \tag{1-1}$$

The dipolar interaction constant in not of much consequence unless users wish to calculate related quantities. For their sake it will now be explicitly calculated for two protons 1Å apart[1], hopefully alleviating any problems in untangling the required unit conversions. Using the values $h = 2\pi h = 1.05459 \times 10^{-34}$ J-sec and $\gamma_{\text{proton}} = 2.675 \times 10^{8}\,\text{sec}^{-1}T^{-1}$ we have

1. The GAMMA defined dipolar interactions constant is slightly different from the commonly used dipolar interaction constant. In frequency units, the latter is defined as $\omega_{ij}^{D} = \frac{h\gamma_i\gamma_j}{r_{ij}^3}$ (neglecting the factor $\frac{\mu_0}{4\pi}$).

$$\xi^D_{HH}\Big|_{1A^\circ} = -2\sqrt{\frac{6\pi}{5}}\frac{\mu_0}{4\pi}\frac{h\gamma_H\gamma_H}{(1A^\circ)^3} = -\sqrt{\frac{6\pi}{5}}\frac{\mu_0}{2\pi}\left[\frac{(1.05459\times10^{-34}\text{J-sec})(2.675\times10^{8}\text{sec}^{-1}T^{-1})^2}{10^{-30}\text{m}^3}\right]$$

$$= -\sqrt{3.76991}\frac{\mu_0}{2\pi}(1.055\times10^{-4}\text{J-sec m}^{-3})(7.156\times10^{16}\text{sec}^{-2}T^{-2})$$

$$= -1.942\frac{\mu_0}{2\pi}(7.546\times10^{12}J\text{sec}^{-1}\text{m}^{-3}T^{-2}) = \frac{-\mu_0}{2\pi}(1.465\times10^{13}J\text{sec}^{-1}\text{m}^{-3}T^{-2})$$

Now substituting in the equalities $\mu_0 = 4\pi\times10^{-7}\text{J-sec}^2\text{C}^{-2}\text{m}^{-1}$ and $1\text{T} = 1\text{JC}^{-1}\text{sec-m}^{-2}$

$$\xi^D_{HH}\Big|_{1A^\circ} = (-2\times10^{-7}\text{J}^1\text{sec}^2C^{-2}m^{-1})(1.465\times10^{13}\text{J}^{-1}\text{C}^2\text{sec}^{-3}\text{-m})$$

$$\xi^D_{HH}\Big|_{1A^\circ} = -2.93\times10^{6}\text{sec}^{-1}$$

Since all spin isotopes (except tritium) have a smaller gyromagnetic ratio than protons and most atoms will be farther apart than 1Å, most likely $\left|\xi^D_{ij}\right| < 2.93\times10^{6}\text{sec}^{-1}$. As distance increases this value drops dramatically, for example at 2Å, $\xi^D_{HH} = -3.664\times10^{5}\text{sec}^{-1}$. This is shown in the following figure for some typical spin pairs[1].

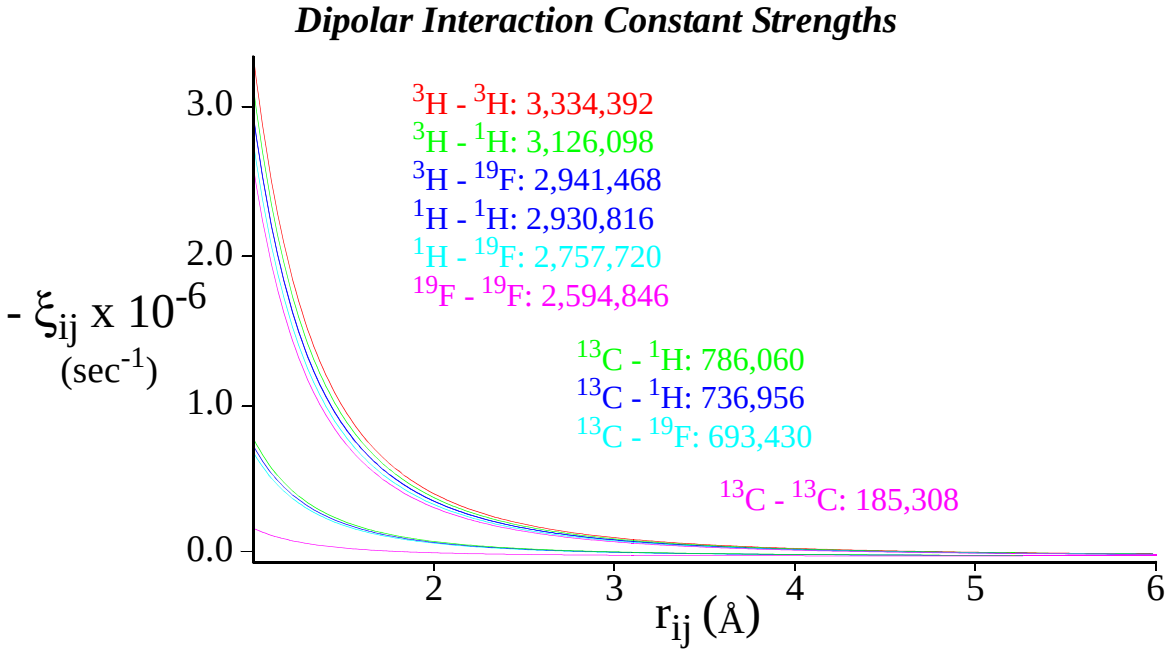


Figure 0-1 **Size of the dipolar interaction constant versus distance for various spin pairs. The values at 1 A are indicated next to the spin pair label.**

1. The GAMMA program which produced this plot is given at the end of this Chapter called Xi_Dip.cc, page 33

2.6.2 Dipole-Dipole Spin-Lattice Relaxation

We now consider the spin lattice or T_1 relaxation expected from the dipolar coupling of two spins. There exist two equations typically found in the literature[1]. The first applies to two unlike spins (superscript DDU - different chemical shifts) where spin I is relaxed by spin S.

$$R_1^{DDU} = \frac{1}{T_1^{DDU}} = \gamma_i^2\gamma_j^2\left[\frac{\mu_0}{4\pi}\right]^2 h^2 S(S+1)\left[\frac{1}{12}J_0(\omega_I-\omega_S)+\frac{3}{2}J_1(\omega_I)+\frac{3}{4}J_2(\omega_I+\omega_S)\right] \quad (1\text{-}2)$$

The second equation applies to two like spins (superscript DDL - equivalent isotope types)

$$R_1^{DDL} = \frac{1}{T_1^{DDL}} = \frac{3}{2}\gamma_i^4\left[\frac{\mu_0}{4\pi}\right]^2 h^2 I(I+1)[J_1(\omega_I)+J_2(2\omega_I)] \quad (1\text{-}3)$$

Both of these equations contain the dipolar power spectral density functions which are listed below[2] for the dynamical case of a spherical top undergoing random rotational motion.

$$J_0(\omega) = \frac{24}{15r^6}[\tau/(1+(\omega_I\tau)^2)] = 6J_1(\omega)$$
$$J_1(\omega) = \frac{4}{15r^6}[\tau/(1+(\omega_I\tau)^2)] \qquad J_2(\omega) = \frac{16}{15r^6}[\tau/(1+(\omega_I\tau)^2)] = 4J_1(\omega) \quad (1\text{-}4)$$

Upon substitution of the above $J(\omega)$ equations we obtain

$$R_1^{DDU} = h^2\gamma_i^2\gamma_j^2\left[\frac{\mu_0}{4\pi}\right]^2\frac{\tau}{15r^6}S(S+1)\left[\frac{2}{1+(\omega_I-\omega_S)^2\tau^2}+\frac{6}{1+(\omega_I\tau)^2}+\frac{12}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_1^{DDL} = h^2\gamma_i^4\left[\frac{\mu_0}{4\pi}\right]^2\frac{2\tau}{5r^6}I(I+1)\left[\frac{1}{1+(\omega_I\tau)^2}+\frac{4}{1+(2\omega_I\tau)^2}\right]$$

As a last step we substitute in the dipolar interaction constant used in GAMMA,

$$\xi_{ij}^D = -2\sqrt{\frac{6\pi}{5}}\left(\frac{\mu_0}{4\pi}\right)h\frac{\gamma_i\gamma_j}{r_{ij}^3}$$

1. See *Pulse and Fourier Transform NMR* by Thomas C. Farrar and Edwin D. Becker, Academic Press, New York, New York, 1971. Specifically the two equations are 4.14 and 4.15 on page 55. Also see "Calculation of Nuclear Spin Relaxation Times" by James L. Sudmeier, *et. al.*, *Conc. Magn. Reson.*, **1990**, 2, 197-212, specifically page 198, equations [3] and [6].
2. These can be found in the previously referenced text on page 51, equations 4.9 as well as in EBW on page 56 (although they leave out the distance to the sixth power factor)

and for working GAMMA equations we obtain

$$R_1^{DDU} = \frac{\xi^2\tau}{72\pi} S(S+1)\left[\frac{2}{1+(\omega_I-\omega_S)^2\tau^2} + \frac{6}{1+(\omega_I\tau)^2} + \frac{12}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$
$$R_1^{DDL} = \frac{\xi^2\tau}{12\pi} I(I+1)\left[\frac{1}{1+(\omega_I\tau)^2} + \frac{4}{1+(2\omega_I\tau)^2}\right] \quad (1\text{-}5)$$

Using these two longitudinal relaxation equations we can easily see how the correlation time affects dipolar T_1 times. The following figure was generated by a GAMMA program[1] for a two spin system 2Å apart. The scalar coupling between the spins set to zero and the spectrometer field strength put at 500 MHz. Keep in mind that this simple picture assumes that the system moves as a spherical top with isotropic motion.

1. This program is listed at the end of this chapter under the name T1plot_Dip.cc, page 34.

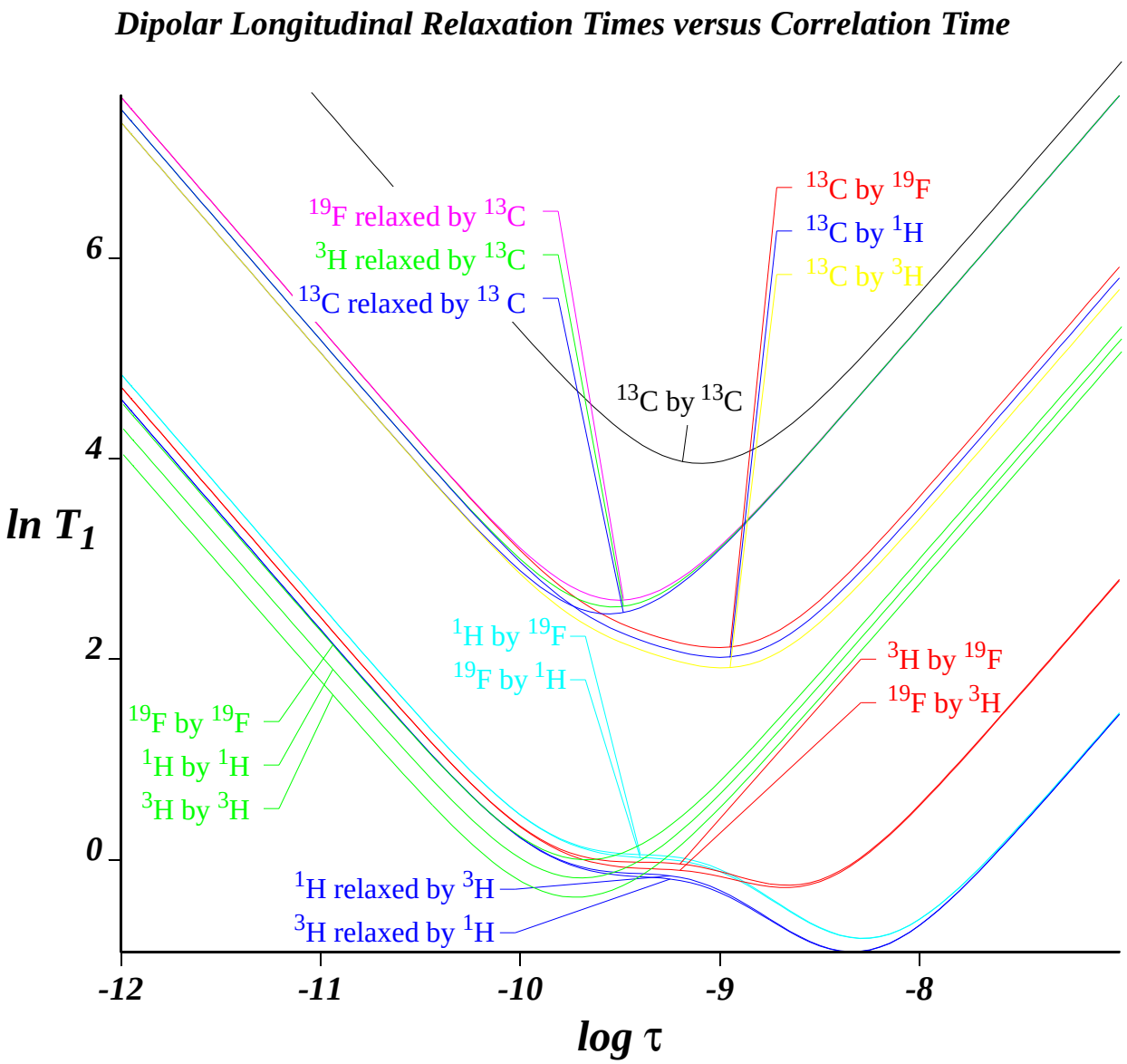


Figure 0-2 **Dipolar longitudinal relaxation times for several spin pairs versus correlation time. The distance between the spins was kept at 2A and the field strength set to 500 MHz.**

We can easily estimate the spin lattice rates under extreme narrowing (EN) conditions. Recall that for extreme narrowing $\omega\tau \ll 1$. In this instance our equations (1-5) become

$$R_1^{DDU}\Big|_{EN} = \frac{20\xi^2\tau}{72\pi}S(S+1) \Rightarrow \frac{5\xi^2\tau}{24\pi} \qquad R_1^{DDL}\Big|_{EN} = \frac{5\xi^2\tau}{12\pi}I(I+1) \Rightarrow \frac{5\xi^2\tau}{16\pi} \tag{1-6}$$

Obviously, the dipolar relaxation rates in extreme narrowing are directly proportional to the correlation time. Using the ^{1}H-^{1}H dipolar interaction constant at 1 A, $\xi_{HH}^D\Big|_{1A^\circ} = -2.93 \times 10^6 \text{sec}^{-1}$, and a correlation time of 1 picosecond, we can directly calculate the spin lattice relaxation time expected in the extreme narrowing limit for the two cases.

$$T_1^{DDU}\Big|_{EN} = \left[\frac{5}{24\pi}\xi^2\tau\right]^{-1} = \left[\frac{5}{24\pi}(-2.93\times10^6\text{sec}^{-1})^2 1.0\times10^{-12}\text{sec}\right]^{-1} = 1.76\text{sec}$$

$$T_1^{DDL}\Big|_{EN} = \left[\frac{5\xi^2\tau}{16\pi}\right]^{-1} = \left[\frac{5}{16\pi}(-2.93\times10^6\text{sec}^{-1})^2 1.0\times10^{-12}\text{sec}\right]^{-1} = 1.17\text{sec}$$

2.6.3 Dipole-Dipole Spin-Spin Relaxation

We now consider the spin-spin or T_2 relaxation expected from the dipolar coupling of two spins. For this simple two spin treatment there exists three equations typically found in the literature[1]. The first applies to two unlike spins (different chemical shifts) where spin I is relaxed by spin S and where there is no resolved scalar coupling between the two spins.

$$R_2^{DDU} = \frac{1}{T_2^{DDU}} = \gamma_i^2\gamma_j^2\left[\frac{\mu_0}{4\pi}\right]^2 h^2 S(S+1)\left[\frac{1}{6}J_0(0) + \frac{1}{24}J_0(\omega_I-\omega_S) + \frac{3}{4}J_1(\omega_I) + \frac{3}{2}J_1(\omega_S) + \frac{3}{8}J_2(\omega_I+\omega_S)\right]$$

If there is resolved scalar coupling between the two unlike spins then the rate at which spin I is relaxed by spin S is given by the following equation.

$$R_2^{DDJ} = \frac{1}{T_2^{DDJ}} = \gamma_i^2\gamma_j^2\left[\frac{\mu_0}{4\pi}\right]^2 h^2 S(S+1)\left[\frac{1}{6}J_0(0) + \frac{1}{24}J_0(\omega_I-\omega_S) + \frac{3}{4}J_1(\omega_I) + \frac{3}{4}J_1(\omega_S) + \frac{3}{8}J_2(\omega_I+\omega_S)\right]$$

The third equation applies to two like spins (same chemical shift and isotope type)

$$R_2^{DDL} = \frac{1}{T_2^{DDL}} = \gamma_i^4\left[\frac{\mu_0}{4\pi}\right]^2 h^2 I(I+1)\left[\frac{3}{8}J_0(0) + \frac{15}{4}J_1(\omega_I) + \frac{3}{8}J_2(2\omega_I)\right]$$

Assuming that the dipole exhibits spherical top random rotational motion, we may substitute in the power dipolar spectral densities given in equation (1-4) on page 20.

1. Two of the three equations can be found in the previous reference, *Pulse and Fourier Transform NMR* by Thomas C. Farrar and Edwin D. Becker, Academic Press, New York, New York, 1971. Specifically the two equations are 4.16 on page 55 and 4.19 on page 56. In the article “Calculation of Nuclear Spin Relaxation Times” by J.L. Sudmeier, S.E. Anderson, and J.S. Frye, *Conc. Magn. Reson.*, **1990**, 2, 197-212, this corresponds to pages 198 and 199, equations [4] and [11]. The third equation, that for two unlike spins with resolved scalar coupling, can be found in EBW on page 504.

$$R_2^{DDU} = \frac{1}{T_2^{DDU}} = \gamma_i^2\gamma_j^2\left[\frac{\mu_0}{4\pi}\right]^2 h^2\frac{\tau}{15r^6}S(S+1)\left[4 + \frac{1}{1+(\omega_I-\omega_S)^2\tau^2} + \frac{3}{1+(\omega_I\tau)^2} + \frac{6}{1+(\omega_S\tau)^2} + \frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_2^{DDJ} = \frac{1}{T_2^{DDJ}} = \gamma_i^2\gamma_j^2\left[\frac{\mu_0}{4\pi}\right]^2 h^2\frac{\tau}{15r^6}S(S+1)\left[4 + \frac{1}{1+(\omega_I-\omega_S)^2\tau^2} + \frac{3}{1+(\omega_I\tau)^2} + \frac{3}{1+(\omega_S\tau)^2} + \frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_2^{DDL} = \frac{1}{T_2^{DDL}} = \gamma_i^4\left[\frac{\mu_0}{4\pi}\right]^2 h^2\frac{\tau}{5r^6}I(I+1)\left[3 + \frac{5}{1+(\omega_I\tau)^2} + \frac{2}{1+(2\omega_I\tau)^2}\right]$$

As a last step we substitute in the dipolar interaction constant used in GAMMA,

$$-\frac{1}{2}\sqrt{\frac{5}{6\pi}}\xi_{ij}^D = \left(\frac{\mu_0}{4\pi}\right)h\frac{\gamma_i\gamma_j}{r_{ij}^3}$$

For the working GAMMA equations we obtain

$$R_2^{DDU} = \frac{1}{T_2^{DDU}} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[4 + \frac{1}{1+(\omega_I-\omega_S)^2\tau^2} + \frac{3}{1+(\omega_I\tau)^2} + \frac{6}{1+(\omega_S\tau)^2} + \frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_2^{DDJ} = \frac{1}{T_2^{DDJ}} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[4 + \frac{1}{1+(\omega_I-\omega_S)^2\tau^2} + \frac{3}{1+(\omega_I\tau)^2} + \frac{3}{1+(\omega_S\tau)^2} + \frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right] \quad (1\text{-}7)$$

$$R_2^{DDL} = \frac{1}{T_2^{DDL}} = \frac{\xi^2\tau}{24\pi}I(I+1)\left[3 + \frac{5}{1+(\omega_I\tau)^2} + \frac{2}{1+(2\omega_I\tau)^2}\right]$$

Using these three equations we can plot the transverse relaxation times, T_2, relative to correlation times. The following figure was generated by a GAMMA program[1] for a two spin system 2Å apart. The scalar coupling between the spins set to zero and the spectrometer field strength put at 500 MHz.

1. This program is listed at the end of this Chapter under the name T2plot_Dip.cc, page 35.

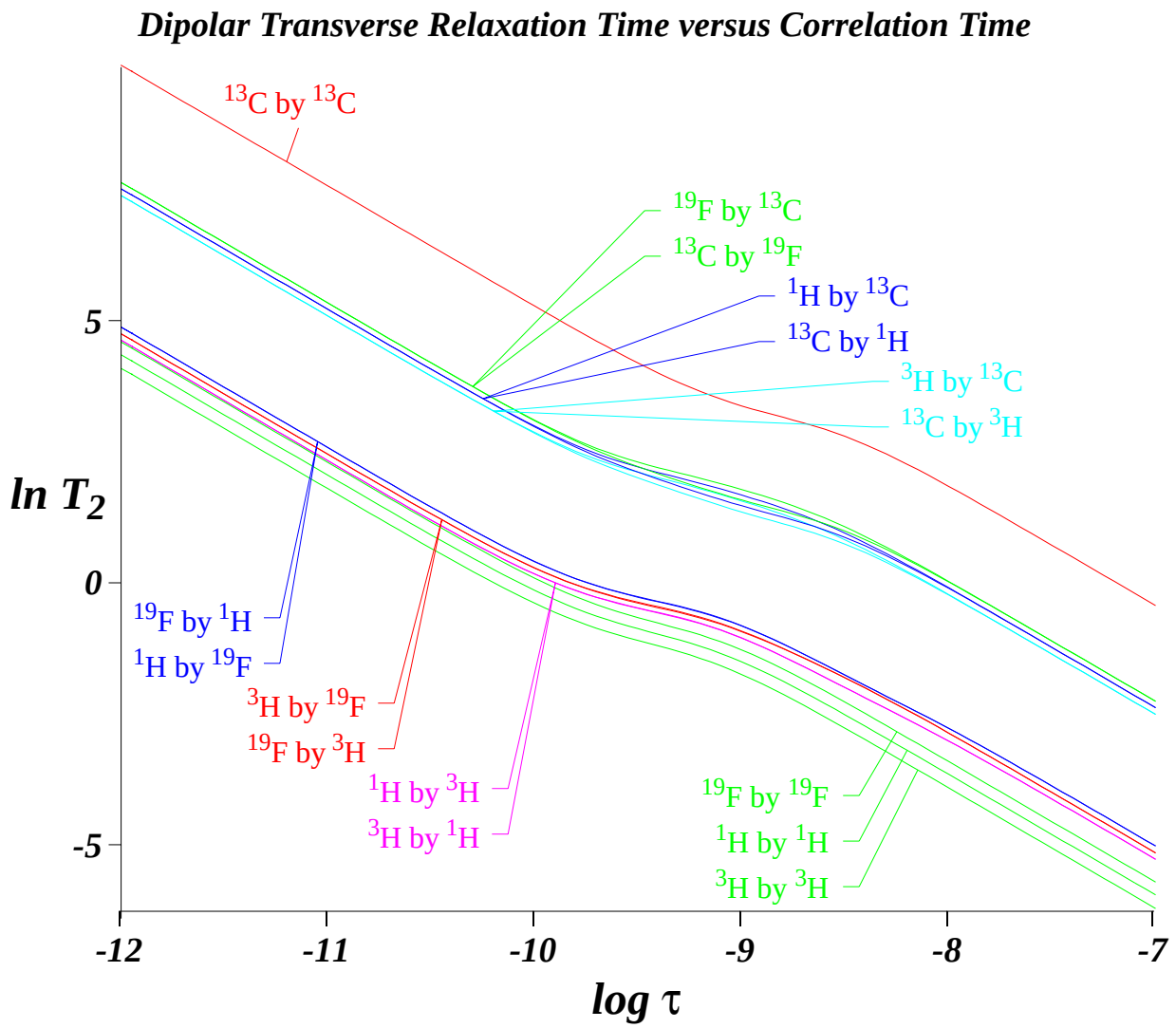


Figure 0-3 **The dipolar transverse relaxation time expected from a single spin pair. The field strength was 500 MHz and the distance between spins 2A, and J=0 Hz.**

We can also estimate the spin-spin rates under extreme narrowing conditions. Recall that for extreme narrowing $\omega\tau \ll 1$. In this instance our equations (1-5) become

$$R_2^{DDU}\Big|_{EN} = \frac{20\xi^2\tau}{72\pi}S(S+1) \Rightarrow \frac{5\xi^2\tau}{24\pi} \qquad R_2^{DDJ}\Big|_{EN} = \frac{17\xi^2\tau}{72\pi}S(S+1) \Rightarrow \frac{17\xi^2\tau}{96\pi}$$

$$R_2^{DDL}\Big|_{EN} = \frac{5\xi^2\tau}{12\pi}I(I+1) \Rightarrow \frac{5\xi^2\tau}{16\pi}$$

Using the proton-proton dipolar interaction constant at 1 Angstrom, $\xi_{HH}^{D}\Big|_{1A^\circ} = 1.465 \times 10^6 \text{sec}^{-1}$, and a correlation time of 1 picosecond we can directly calculate the spin lattice relaxation time expected in the extreme narrowing limit for all three cases.

$$T_2\big|_{AX} = \left[\frac{5}{24\pi}\xi^2\tau\right]^{-1} = \left[\frac{5}{24\pi}(-2.93\times10^6\text{sec}^{-1})^2 1.0\times10^{-12}\text{sec}\right]^{-1} = 1.76\text{sec}$$

$$T_2\Big|_{\substack{AX \\ \text{Resolved J}}} = \left[\frac{17}{96\pi}\xi^2\tau\right]^{-1} = \left[\frac{17}{96\pi}(-2.93\times10^6\text{sec}^{-1})^2 1.0\times10^{-12}\text{sec}\right]^{-1} = 2.07\text{sec}$$

$$T_2\big|_{A_2} = \left[\frac{5}{16\pi}\xi^2\tau\right]^{-1} = \left[\frac{5}{16\pi}(-2.93\times10^6\text{sec}^{-1})^2 1.0\times10^{-12}\text{sec}\right]^{-1} = 1.17\text{sec}$$

Notice that in the extreme narrowing limit $T_1 = T_2$ for dipolar relaxation between two spins.

One may also obtain formulae for the transverse relaxation times of the multiple quantum transitions in a two spin system. For dipolar relaxation with isotropic random motion equations taken from the literature[1] in the AX case with resolved J coupling are the following[2].

$$[R_2^{DDJ}]^{ZQT} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[\frac{2}{1+[(\omega_I-\omega_S)\tau]^2}+\frac{3}{1+(\omega_I\tau)^2}+\frac{3}{1+(\omega_S\tau)^2}\right]$$

$$[R_2^{DDJ}]^{SQT} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[4+\frac{1}{1+\Delta\omega_{IS}^2\tau^2}+\frac{3}{1+(\omega_I\tau)^2}+\frac{3}{1+(\omega_S\tau)^2}+\frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$[R_2^{DDJ}]^{DQT} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[\frac{3}{1+(\omega_I\tau)^2}+\frac{3}{1+(\omega_S\tau)^2}+\frac{12}{1+[(\omega_I+\omega_S)\tau]^2}\right]$$

We have of course already seen the equation for transverse relaxation of single quantum transitions (SQT). In the extreme narrowing limit these equations become (for two spin 1/2 particles)

$$[R_2^{DDJ}]^{ZQT}\Big|_{EN} = \frac{8\xi^2\tau}{72\pi}S(S+1) \Rightarrow \frac{\xi^2\tau}{12\pi} \qquad [R_2^{DDJ}]^{SQT}\Big|_{EN} = \frac{17\xi^2\tau}{72\pi}S(S+1) \Rightarrow \frac{17\xi^2\tau}{96\pi}$$

Now

$$[R_2^{DDJ}]^{DQT}\Big|_{EN} = \frac{18\xi^2\tau}{72\pi}S(S+1) \Rightarrow \frac{3\xi^2\tau}{16\pi}$$

we estimate the relaxation times expected for the multiple quantum transitions of two protons an Angstrom apart in the extreme narrowing limit.

$$[T_2^{DDJ}]^{ZQT} = \left[\frac{\xi^2\tau}{12\pi}\right]^{-1} = \left[\frac{1}{12\pi}(-2.93\times10^6\text{sec}^{-1})^2 1.0\times10^{-12}\text{sec}\right]^{-1} = 4.39\text{sec}$$

$$[T_2^{DDJ}]^{DQT} = \left[\frac{3\xi^2\tau}{16\pi}\right]^{-1} = \left[\frac{3}{16\pi}(-2.93\times10^6\text{sec}^{-1})^2 1.0\times10^{-12}\text{sec}\right]^{-1} = 1.95\text{sec}$$

$$[T_2^{DDJ}]^{SQT} = \left[\frac{17\xi^2\tau}{96\pi}\right]^{-1} = \left[\frac{17}{96\pi}(-2.93\times10^6\text{sec}^{-1})^2 1.0\times10^{-12}\text{sec}\right]^{-1} = 2.07\text{sec}$$

1. See Ernst, Bodenhausen, and Wokaun, page 504, below equation (9.4.7).
2. For the A2 system or AX (J=0), multiple quantum coherence does not develop so there are no analogous equations for the DDL and DDU cases. Furthermore, there is no contribution to longitudinal relaxation from these transitions so that there are no analogous formulae involving T_1.

Bear in mind that these equations are for two unlike spins (differing chemical shifts) with resolved scalar coupling values. For two spins with unresolved coupling one can ascertain the analogous formulas for the single quantum transition transverse relaxation from Abragam's equations[1].

2.6.4 Dipole-Dipole Relaxation Linewidths

The linewidths expected from dipolar relaxation may be estimated directly from the spin-spin relaxation times according to the following relationship.

$$LW_{hh}^{DD} = R_2^{DD}/\pi \tag{1-8}$$

Here LW_{hh}^{DD} is used for the line-width at half-height.

2.6.5 Two Spin Approximation

In a multiple spin system one may apply the two spin approximation and assume that the total relaxation rate is the sum over all rates from spin pairs. Thus we have the following equation for the spin-lattice and spin-spin relaxation rates of spin i.

$$\frac{1}{T_1^{DD}(i)} = R_1^{DD}(i) = \sum_{i>i}^{spins} R_1^{DD}(i,j) \qquad \frac{1}{T_2^{DD}(i)} = R_2^{DD}(i) = \sum_{i>i}^{spins} R_2^{DD}(i,j) \tag{1-9}$$

The linewidth for a transition associated with a particular spin would depend on the R_2 estimated using the two spin approximation.

$$LW_{hh}^{DD}(i) = R_2^{DD}(i)/\pi = \left(\sum_{j>i}^{spins} R_2^{DD}(i,j) \right)/\pi$$

2.6.6 Nuclear Overhauser Effect (NOE)

Consider the spin, i, being relaxed by another spin, j, through a dipolar interaction. Were one to irradiate at the resonance frequency of spin j for a time such that a steady state is attained the magnetization of spin i will be altered due to a redistribution of the Boltzmann populations. This is called the Nuclear Overhauser Effect (NOE) and placed in terms of the fractional enhancement of the magnetization, ρ[2].

$$\rho_i = \frac{\gamma_j \left[\frac{6}{1+(\omega_i+\omega_j)^2\tau^2} - \frac{1}{1+\Delta\omega_{ij}^2\tau^2} \right]}{\gamma_i \left[\frac{1}{1+\Delta\omega_{ij}^2\tau^2} + \frac{3}{1+(\omega_i\tau)^2} + \frac{6}{1+(\omega_i+\omega_j)^2\tau^2} \right]} \tag{1-10}$$

1. See Abragam, page 296, equation (89) for the case of unlike spins and see page 292, equation (79) and the discussion in text on page 297 focused on equation (90) for the treatment of equivalent spins.
2. The equations can be found in the article "Calculation of Nuclear Spin Relaxation Times" by J.L. Sudmeier, S.E. Anderson, and J.S. Frye, *Conc. Magn. Reson.*, **1990**, 2, 197-212, this corresponds to pages 200 and 201.

Keep in mind that this formula provides the fractional enhancement to spin i which is being relaxed through a dipolar interaction by spin j. When $\rho_i = 0.5$ there will be an observed enhancement of 50% in the integrated intensity of spin i. When $\rho_i = -1.0$ there will be complete disappearance of transitions associated with the spin. Maximum NOE enhancement occurs in the extreme narrowing limit, when $\omega\tau \ll 1$. Under such conditions we have

$$\rho_i(max) = \rho_i\big|_{EN} = \frac{0.5\gamma_j}{\gamma_i} \tag{1-11}$$

In the opposite motional regime, when $\omega\tau \gg 1$, the minimum NOE is observed (assuming the gyromagnetic ratios are the same sign).

$$\rho_i(min) = \rho_i\big|_{EB} = \frac{-\gamma_j}{\gamma_i} \tag{1-12}$$

This implies that for large molecules (proteins), homonuclear decoupling can cause proton resonances to disappear completely. NOE values are also commonly reported in terms of η_{NOE} where

$$\eta_{NOE} = 1 + \rho$$

and then

$$\eta_{NOE}(max) = 1 + \frac{\gamma_j}{2\gamma_i}$$

2.6.7 Dipole-Dipole Relaxation Equations

We now group together the important equations regarding a the simple treatment of dipolar relaxation.

Dipolar Relaxation Equations

Interaction Constant

$$\xi_{ij}^{D} = -2\sqrt{\frac{6\pi}{5}}\left(\frac{\mu_0}{4\pi}\right)h\frac{\gamma_i\gamma_j}{r_{ij}^3}$$

Longitudinal Relaxation (Spin-Lattice)

$$R_1^{DDU} = \frac{1}{T_1^{DDU}} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[\frac{2}{1+\Delta\omega_{IS}^2\tau^2} + \frac{6}{1+(\omega_I\tau)^2} + \frac{12}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_1^{DDL} = \frac{1}{T_1^{DDU}} = \frac{\xi^2\tau}{12\pi}I(I+1)\left[\frac{1}{1+(\omega_I\tau)^2} + \frac{4}{1+(2\omega_I\tau)^2}\right]$$

$$R_1^{DDU} \xrightarrow[\textbf{Narrowing}]{\textbf{Extreme}} \frac{5\xi^2\tau}{24\pi} \qquad R_1^{DDL} \xrightarrow[\textbf{Narrowing}]{\textbf{Extreme}} \frac{5\xi^2\tau}{16\pi}$$

Transverse Relaxation (Spin-Spin)

$$R_2^{DDU} = \frac{1}{T_2^{DDU}} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[4 + \frac{1}{1+\Delta\omega_{IS}^2\tau^2} + \frac{3}{1+\omega_I^2\tau^2} + \frac{6}{1+\omega_S^2\tau^2} + \frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_2^{DDJ} = \frac{1}{T_2^{DDJ}} = \frac{\xi^2\tau}{72\pi}S(S+1)\left[4 + \frac{1}{1+\Delta\omega_{IS}^2\tau^2} + \frac{3}{1+\omega_I^2\tau^2} + \frac{3}{1+\omega_S^2\tau^2} + \frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$R_2^{DDL} = \frac{1}{T_2^{DDL}} = \frac{\xi^2\tau}{24\pi}I(I+1)\left[3 + \frac{5}{1+(\omega_I\tau)^2} + \frac{2}{1+(2\omega_I\tau)^2}\right]$$

$$R_2^{DDU} \xrightarrow[\textbf{I = 1/2}]{\textbf{EN}} \frac{5}{24\pi}\xi^2\tau \qquad R_2^{DDL} \xrightarrow[\textbf{I = 1/2}]{\textbf{EN}} \frac{17}{96\pi}\xi^2\tau \qquad R_2^{DDL} \xrightarrow[\textbf{I = 1/2}]{\textbf{EN}} \frac{5}{16\pi}\xi^2\tau$$

Dipolar Relaxation Equations

Multiple Quantum Relaxation

$$[R_2^{DDJ}]^{ZQT} = \frac{\xi^2\tau}{72\pi} S(S+1)\left[\frac{2}{1+[(\omega_I-\omega_S)\tau]^2} + \frac{3}{1+(\omega_I\tau)^2} + \frac{3}{1+(\omega_S\tau)^2}\right]$$

$$[R_2^{DDJ}]^{SQT} = \frac{\xi^2\tau}{72\pi} S(S+1)\left[4 + \frac{1}{1+\Delta\omega_{IS}^2\tau^2} + \frac{3}{1+(\omega_I\tau)^2} + \frac{3}{1+(\omega_S\tau)^2} + \frac{6}{1+(\omega_I+\omega_S)^2\tau^2}\right]$$

$$[R_2^{DDJ}]^{DQT} = \frac{\xi^2\tau}{72\pi} S(S+1)\left[\frac{3}{1+(\omega_I\tau)^2} + \frac{3}{1+(\omega_S\tau)^2} + \frac{12}{1+[(\omega_I+\omega_S)\tau]^2}\right]$$

$$[R_2^{DDJ}]^{ZQT} \xrightarrow[\boldsymbol{I = 1/2}]{\boldsymbol{EN}} \frac{8}{96\pi}\xi^2\tau_c \qquad [R_2^{DDJ}]^{SQT} \xrightarrow[\boldsymbol{I = 1/2}]{\boldsymbol{EN}} \frac{17}{96\pi}\xi^2\tau_c \qquad [R_2^{DDJ}]^{DQT} \xrightarrow[\boldsymbol{I = 1/2}]{\boldsymbol{EN}} \frac{9}{96\pi}\xi^2\tau_c$$

LineWidth at Half Height

$$LW_{hh}^{DD} = R_2^{DD}/\pi$$

Two Spin Approximation

$$\frac{1}{T_1^{DD}(i)} = R_1^{DD}(i) = \sum_{j>i}^{spins} R_1^{DD}(i,j) \qquad \frac{1}{T_2^{DD}(i)} = R_2^{DD}(i) = \sum_{j>i}^{spins} R_2^{DD}(i,j)$$

$$LW_{hh}^{DD}(i) = R_2^i/\pi = \left(\sum_{\substack{j \\ i\neq j}}^{spins} R_2^{ij}\right)/\pi$$

2.6.8 Dipole-Dipole Two Spin Relaxation

At this point we apply our formulae for the dipolar relaxation in a two spin system. The following tables tabulate the various relaxation values expected for a two spin system at 500 MHz for different internuclear distances and correlation times[1]. Please note that the T_1 listed for resolved scalar

coupling are not computed differently that those without scalar coupling for this table.

Table 1: Estimated Proton Dipolar Relaxation Times @ 500 MHz

r (Å)	tau (sec)	T_1 (sec)	T_2 (sec)	LW (Hertz)	T_1 (sec)	T_2 (sec)	LW (Hertz)	T_1 (sec)	T_2 (sec)	LW (Hertz)
		^{1}H by ^{1}H			^{1}H by ^{13}C Unresolved J			^{1}H by ^{13}C Resolved J		
1	10^{-12}	1.17041	1.17038	0.27197	27.7660	32.6657	.009744	27.7660	27.7659	.011464
	10^{-11}	0.11743	.117187	2.71626	2.78012	3.26904	.097371	2.78012	2.77839	.114566
	10^{-10}	.015491	.013027	24.4351	.312858	.350218	.908892	.312858	.294789	1.07979
	10^{-9}	.030681	.003335	95.4406	.349818	.083619	3.80666	.349818	.065422	4.86552
	10^{-8}	.289099	.000389	817.435	3.18386	.013691	23.2495	3.18386	.013533	23.5207
	10^{-7}	2.88917	3.90e-05	8159.37	31.8066	.001388	229.316	31.8066	.001388	229.343
2	10^{-12}	74.9061	74.9046	0.00425	1777.03	2090.61	.000152	1777.03	1777.01	.000179
	10^{-11}	7.5155	7.49996	.042442	177.927	209.219	.001521	177.927	177.817	0.00179
	10^{-10}	.991427	0.83371	.381799	20.0229	22.4139	.014201	20.0229	18.8665	.016872
	10^{-9}	1.96359	0.21345	1.49126	22.3883	5.35163	.059479	22.3883	4.18698	.076024
	10^{-8}	18.5023	.024922	12.7724	203.767	.876226	.363274	203.767	.866122	.367511
	10^{-7}	184.907	.002497	127.490	2035.62	.088838	3.58306	2035.62	.088827	3.58349
3	10^{-12}	853.228	853.210	.000373	20241.4	23813.3	1.337e-05	20241.4	20241.3	1.573e-05
	10^{-11}	85.6062	85.4293	.003726	2026.71	2383.13	.000134	2026.71	2025.45	.000157
	10^{-10}	11.2930	9.49648	.033519	228.074	255.309	.001247	228.074	214.901	.001481
	10^{-9}	22.3665	2.43133	0.13092	255.017	60.9584	.005222	255.017	47.6923	.006674
	10^{-8}	210.753	.283873	1.12131	2321.03	9.98076	.031892	2321.03	9.86567	.032264
	10^{-7}	2106.21	.028439	11.1925	23187.0	1.01191	.314562	23187.0	1.01179	0.3146
4	10^{-12}	4793.99	4793.89	6.64e-05	113730	133799	2.379e-06	113730	113729	2.799e-06
	10^{-11}	480.992	479.998	.000663	11387.4	13390.0	2.377e-05	11387.4	11380.3	2.797e-05
	10^{-10}	63.4513	53.3575	005966	1281.47	1434.49	.000222	1281.47	1207.46	.000264
	10^{-9}	125.670	13.6608	.023301	1432.85	342.504	.000929	1432.85	267.967	.001188
	10^{-8}	1184.15	1.59499	.199569	13041.1	56.0785	.005676	13041.1	55.4318	.005742
	10^{-7}	11834.1	.159791	1.99203	130280	5.68560	.055985	130280	5.68492	.055992
5	10^{-12}	18287.6	18287.2	1.74e-05	433844	510402	6.236e-07	433844	433841	7.337e-07
	10^{-11}	1834.84	1831.05	.000174	43439.3	51078.8	6.232e-06	43439.3	43412.4	7.332e-06
	10^{-10}	242.047	203.543	.001564	4888.41	5472.15	5.817e-05	4888.41	4606.08	6.911e-05
	10^{-9}	479.392	52.1119	.006108	5465.90	1306.55	.000244	5465.90	1022.21	.000311
	10^{-8}	4517.16	6.08439	.052316	49747.8	213.922	.001488	49747.8	211.456	.001505
	10^{-7}	45121.4	.609556	0.5222	496979	21.6888	.014676	496979	21.6862	.014678

Evidently, protons are much more effective at relaxing protons than are carbons. This is seen immediately from the calculated linewidths, those in the carbon columns being much smaller than those in the proton column. Comparison of the values with and without scalar coupling reveals that when scalar coupling is present relaxation occurs more readily than when there is none. The calculated linewidths for the column with resolved J are larger than those without J coupling. Keep in mind that the greatest difference in the times shown result from differences in the gyromagnetic rations. Since carbon and proton ratios are different by roughly a factor of 4, we roughly see this

1. This table and others like it may be produced from the GAMMA program T1T2_Dip.cc provided at the end of this Chapter, page 36.

factor in comparing proton-proton versus proton-carbon relaxation.

We can compare these proton relaxation values with carbon relaxation values. Under similar conditions, carbon relaxes carbon much more slowly than proton-proton relaxation. On the other hand, cross relaxation rates are nearly identical.

Table 2: Estimated Carbon Dipolar Relaxation Times @ 500 MHz

r (Å)	tau (sec)	T_1 (sec)	T_2 (sec)	LW (Hertz)	T_1 (sec)	T_2 (sec)	LW (Hertz)	T_1 (sec)	T_2 (sec)	LW (Hertz)
		^{13}C by ^{13}C			^{13}C by ^{1}H Unresolved J			^{13}C by ^{1}H Resolved J		
1	10^{-12}	292.760	292.760	.001087	27.7659	32.6657	.009744	27.7659	27.7659	.011464
	10^{-11}	29.2822	29.2783	.010872	2.77935	3.26904	.097371	2.77935	2.77878	0.11455
	10^{-10}	2.98955	2.95112	.107861	.304257	.350218	.908892	.304257	.298769	1.06541
	10^{-9}	.831761	.440185	.723127	.117404	.083619	3.80666	.117404	.080283	3.96486
	10^{-8}	4.61285	.094841	3.35624	.523875	.013691	23.2495	.523875	.013681	23.2669
	10^{-7}	45.6773	.009756	32.6278	5.16505	.001388	229.316	5.16505	.001388	229.317
2	10^{-12}	18736.6	18736.6	1.699e-05	1777.02	2090.61	.000152	1777.02	1777.02	.000179
	10^{-11}	1874.06	1873.81	0.00017	177.878	209.219	.001521	177.878	177.842	0.00179
	10^{-10}	191.331	188.872	.001685	19.4724	22.4139	.014201	19.4724	19.1212	.016647
	10^{-9}	53.2327	28.1719	.011299	7.51383	5.35163	.059479	7.51383	5.13809	.061951
	10^{-8}	295.222	6.06983	.052441	33.5280	.876226	.363274	33.5280	.875571	.363546
	10^{-7}	2923.35	0.62437	0.50981	330.563	.088838	3.58306	330.563	.088837	3.58308
3	10^{-12}	213422	213422	1.492e-06	20241.4	23813.3	1.337e-05	20241.4	20241.3	1.573e-05
	10^{-11}	21346.7	21343.9	1.491e-05	2026.14	2383.13	.000134	2026.14	2025.73	.000157
	10^{-10}	2179.38	2151.37	.000148	221.803	255.309	.001247	221.803	217.802	.001461
	10^{-9}	606.354	320.895	.000992	85.5872	60.9584	.005222	85.5872	58.5261	.005439
	10^{-8}	3362.77	69.1392	.004604	381.905	9.98076	.031892	381.905	9.97330	.031916
	10^{-7}	33298.7	7.11197	.044757	3765.32	1.01191	.314562	3765.32	1.01191	.314564
4	10^{-12}	1.20e+06	1.20e+06	2.655e-07	113729	133799	2.380e-06	113729	113729	2.799e-06
	10^{-11}	119940	119924	2.654e-06	11384.2	13390.0	2.377e-05	11384.2	11381.9	2.797e-05
	10^{-10}	12245.2	12087.8	2.633e-05	1246.24	1434.49	.000222	1246.24	1223.76	0.00026
	10^{-9}	3406.89	1803.00	.000177	480.885	342.504	.000929	480.885	328.838	.000968
	10^{-8}	18894.2	388.469	.000819	2145.79	56.0785	.005676	2145.79	56.0365	0.00568
	10^{-7}	187094	39.9597	.007966	21156.1	5.68560	.055985	21156.1	5.68556	.055986
5	10^{-12}	4.57e+06	4.57e+06	6.959e-08	433843	510402	6.237e-07	433843	433842	7.337e-07
	10^{-11}	457534	457474	6.958e-07	43427.3	51078.8	6.232e-06	43427.3	43418.4	7.331e-06
	10^{-10}	46711.7	46111.2	6.903e-06	4754.02	5472.15	5.817e-05	4754.02	4668.26	6.817e-05
	10^{-9}	12996.3	6877.90	4.628e-05	1834.43	1306.55	.000244	1834.43	1254.42	.000254
	10^{-8}	72075.8	1481.89	.000215	8185.55	213.922	.001488	8185.55	213.762	.001489
	10^{-7}	713707	152.434	.002088	80704.0	21.6888	.014676	80704.0	21.6887	.014676

2.7 Dipolar Relaxation Source Codes

Xi_Dip.cc

Generate Plots of Dipolar Interaction Constants

```
/** Xi_Dip.cc
****************************************** c++ **
**                                          **
**                  GAMMA Example Program   **
**                                          **
**************************************************************
*/

#include <gamma.h>
main (int argc, char* argv[])
{
 sys_dynamic dsys(2);                       // Set up a 2 spin system
 int npts = 301;                            // Use 301 points each xi vs. r
 String types[4];                           // Store the isotope types
 types[0] = "1H";
 types[1] = "3H";
 types[2] = "19F";
 types[3] = "13C";
 row_vector plot(npts), plots[10];          // Storage for plots
 coord pt(0,0,0);                           // Set the first spin at 0,0,0
 dsys.put(pt, 0);
 matrix xiDs;                               // Matrix for xi values
 double zinc = 5.0e-10/double(npts-1);      // Increment r 5 Angs. total
 int k=0;
 for(int iso1=0; iso1<4; iso1++)            // Loop through all isotope pairs
  {
  dsys.isotope(0, types[iso1]);             // Get first isotope
  for(int iso2=iso1; iso2<4; iso2++)
    {
    dsys.isotope(1, types[iso2]);           // Get second isotope
    double z = 1.0e-10;                     // Start at 1 Angstrom
     for(int i=0; i<npts; i++)              // Loop through all points
     {
     pt.xyz(0,0,z);                         // Set coordinate of 2nd spin
     dsys.put(pt, 1)
     xiDs = xiD(dsys);                      // Calculate new xi matrix
     plot.put(-1.0*xiDs.get(0,1), i);       // Store new xi value negated
     z += zinc;                             // Increment distances
    }
   plots[k] = plot;                         // Store this xi vs. r plot
   k++;
   }
  }
FM_1Dm("xiDplot.mif",k,plots,19,14,1,6);    // Output all plots to FM
}
```

T1plot_Dip.cc

Generate Plots of Dipolar T1 *versus* tau

```
/* T1plot_Dip.cc ***************************************************************-*-c++-*-
**                                                                               **
**                          Example program for the GAMMA Library                **
**                                                                               **
** This program constructs a plot of T1 versus tau for a two spin system        **
** the effects of dipolar relaxation. The calculations are performed using a    **
** simple analytic formula for the T1 time which assumes that the spin          **
** system motion is that of a spherical top under rotationally diffusive motion. **
**                                                                               **
*********************************************************************************** */

#include <relax_Dip.h>

main (int argc, char* argv[])
 {
 cout << "\n\n\t\t\tGAMMA NMR Checking Program";
 cout << "\n\t\t Dipolar T1 Relaxation - Two Spin System\n\n";

 sys_dynamic dsys(2);                          // Set up a 2 spin system
 int npts = 101;                               // Use 101 points each T1 vs. tau
 String types[4];                              // Store the isotope types
 types[0] = "1H";
 types[1] = "3H";
 types[2] = "19F";
 types[3] = "13C";
 row_vector plot(npts), plots[12];             // Storage for plots
 coord pt(0,0,0);                              // Set the first spin at 0,0,0
 dsys.put(pt, 0);
 pt.xyz(0, 0, 2e-10);                          // Set coordinate of 2nd spin
 dsys.put(pt, 1);
 double bigO;                                  // Set the spectrometer frequency
 query_parameter(argc, argv, 1,
   "Spectrometer Frequency (MHz)? ", bigO);
 dsys.Omega(bigO);
 double lntauinc = 5.0/double(npts-1);         // Increment tau 10**5 sec
 double lntau, tau, T1;
 int k=0;
 for(int iso1=0; iso1<4; iso1++)               // Loop through all isotope pairs
   {
   dsys.isotope(0, types[iso1]);               // Set first isotope
   for(int iso2=0; iso2<4; iso2++)
     {
     dsys.isotope(1, types[iso2]);             // Set second isotope
     tau = 1.0e-12;                            // Start at 1 psec tau
     lntau = -12.0;
     dsys.taux(tau);                           // Set tau value of system
     for(int i=0; i<npts; i++)                 // Loop through all points
       {
       dsys.taux(tau);                         // Set tau value of system
       T1 = T1_DD(dsys, 0);                    // Calculate new T1 value
       if(i == 0)                              // Output inital and final values
         {                                     // so plots are discernable
          cout << "\n" << types[iso1]
               << " - " << types[iso2]
               << ": initial ln(T1) = "
               << log(T1);
         }
       else if(i == npts - 1)
         cout << ": final ln(T1) = "
              << log(T1);
       plot.put(log(T1), i);                   // Store new T1 value
       lntau += lntauinc;                      // Increment log of tau
       tau = pow(10.0, lntau);                 // Determine next tau
       }
     plots[k] = plot;                          // Store this T1 vs. tau plot
     k++;
     }
   }
 FM_1Dm("T1Dplot.mif",k,plots,19,14,-12,-7);   // Output all plots to FrameMaker
 cout << "\n\n";                               // Keep screen looking nice
}
```

T2plot_Dip.cc

Generate Plots of Dipolar T2 *versus* tau

(This program is nearly identical to T1plot_Dip.cc)

```
/* T2plot_Dip.cc ************************************-*-c++-                    *-
**                                                                              **
**
Example program for the GAMMA Library                                           **
**                                                                              **
** This program constructs a plot of T2 versus tau for a two                    **
** spin system under the effects of dipolar relaxation. The                     **
** calculations are performed using a simple analytic formula                   **
** for the T2 time which assumes that the spin system motion                    **
** is that of a spherical top under rotationally diffusive                      **
** motion.                                                                      **
**                                                                              **
*********************************************************************          */

#include <relax_Dip.h>

main (int argc, char* argv[])
 {
 cout << "\n\n\t\t\tGAMMA NMR Checking Program";
 cout << "\n\t\t Dipolar T2 Relaxation - Two Spin System\n\n";

 sys_dynamic dsys(2);                          // Set up a 2 spin system
 int npts = 101;                               // Use 101 points each T2 vs. tau
 String types[4];                              // Store the isotope types
 types[0] = "1H";
 types[1] = "3H";
 types[2] = "19F";
 types[3] = "13C";
 row_vector plot(npts), plots[12];             // Storage for plots
 coord pt(0,0,0);                              // Set the first spin at 0,0,0
 dsys.put(pt, 0);
 pt.xyz(0, 0, 2e-10);                          // Set coordinate of 2nd spin
 dsys.put(pt, 1);
 double bigO;                                  // Set the spectrometer frequency
 query_parameter(argc, argv, 1,
  "Spectrometer Frequency (MHz)? ", bigO);
 dsys.Omega(bigO);
 double lntauinc = 5.0/double(npts-1);         // Increment tau 10**5 sec
 double lntau, tau, T2;
 int k=0;
 for(int iso1=0; iso1<4; iso1++)               // Loop through all isotope pairs
  {
  dsys.isotope(0, types[iso1]);                // Set first isotope
  for(int iso2=0; iso2<4; iso2++)
   {
   dsys.isotope(1, types[iso2]);               // Set second isotope
   tau = 1.0e-12;                              // Start at 1 psec tau
   lntau = -12.0;
   dsys.taux(tau);                             // Set tau value of system
   for(int i=0; i<npts; i++)                   // Loop through all points
    {
    dsys.taux(tau);                            // Set tau value of system
    T2 = T2_DD(dsys, 0);                       // Calculate new T2 value
    if(i == 0)                                 // Output inital and final values
     {                                         // so plots are discernable
      cout << "\n" << types[iso1]
           << " - " << types[iso2]
           << ": initial ln(T2) = "
           << log(T2);
     }
    else if(i == npts - 1)
      cout << ": final ln(T2) = "
           << log(T2);
    plot.put(log(T2), i);                      // Store new T2 value
    lntau += lntauinc;                         // Increment log of tau
    tau = pow(10.0, lntau);                    // Determine next tau
    }
   plots[k] = plot;                            // Store this T2 vs. tau plot
   k++;
   }
  }
 FM_1Dm("T2Dplot.mif",k,plots,19,14,-12,-7);   // Output all plots to FrameMaker
 cout << "\n\n";                               // Keep screen looking nice
}
```

T1T2_Dip.cc

Classical Dipolar Relaxation Values Table

```
/* T1T2_Dip.cc ****************************************-*-c++-                *-
**                                                                          **
**
Example program for the GAMMA Library                                       **
**                                                                          **
** This program performs a simple calculation of the proton                 **
** T1, T2, and linewidth expected from dipolar relaxation in                **
** a two spin system. The calculations are performed using a                **
** simple approximation, the analytical formulas used assume                **
** the spin system motion is that of a spherical top under                  **
** rotationally diffusive motion.                                           **
**                                                                          **
******************************************************************          */

#include <relax_Dip.h>

main (int argc, char* argv[])
 {
 cout << "\n\n\t\t\tGAMMA NMR Checking Program";
 cout << "\n\t\t Dipolar Relaxation - Two Spin System\n\n";

 sys_dynamic dsys(2);                          // Set up a 2 spin system
 coord pt(0,0,0);                              // Set the first spin at 0,0,0
 dsys.put(pt, 0);
 double disti = 1.e-10;                        // Start at 1 Angstrom dipole
 double taui = 1.e-12;                         // Start at 10**-12 correlation
 double tau, dist;                             // Used for the actual values
 double bigO;                                  // Set the spectrometer frequency
 query_parameter(argc, argv, 1,
 "Spectrometer Frequency (MHz)? ", bigO);
 dsys.Omega(bigO);

 String iso0, iso1;                            // Set up the isotope types
 query_parameter(argc, argv, 2,                // Get first isotope
      "Isotope Type For Spin 1? ", iso0);
 dsys.isotope(0, iso0);                        // Set first isotope
 query_parameter(argc, argv, 3,                // Get second isotope
      "Isotope Type For Spin 2? ", iso1);
 dsys.isotope(1, iso1);                        // Set second isotope

 double Jval = 0.0;
 if(iso0 != iso1)
 query_parameter(argc, argv, 4,                // Get scalar coupling
       "Scalar Coupling Value?", Jval);
 dsys.J(0,1,Jval);

 cout << "\n\n\tSpin 1: " << dsys.symbol(0)
 << " Relaxed by Spin 2: " << dsys.symbol(1)
 << "\n\nR\ttau\tT1\tT2\tLWHH";
 dist = disti;
 pt.xyz(0, 0, dist);                           // Set coordinate of 2nd spin
 dsys.put(pt, 1);
 double T1, T2, LW;
 for(int i=1; i<6; i++)                        // Loop over different distances
  {
  dsys.taux(taui);                             // Set initial correlation time
  tau = dsys.taux();
  for(int j=-12; j<-6; j++)                    // Loop over different taus
    {
    cout << "\n" << dist*1.e10
         << "\t" << tau;
    T1 = T1_DD(dsys, 0, 1);                    // Calculate T1
    T2 = T2_DD(dsys, 0, 1);                    // Calculate T2
    LW = 1.0/(T2*PI);                          // Calculate Linewidth from T2
    cout << "\t" << T1 << "\t"
         << T2 << "\t" << LW;
    tau *= 10.0;                               // Increase tau by 10x
    dsys.taux(tau);
    }
  dist += 1.e-10;                              // Increase separation by 1 Angs.
  pt.xyz(0, 0, dist);                          // Reset coordinate of 2nd spin
  dsys.put(pt, 1);
  }
 cout << "\n";                                 // Tidy up output
 }
```

3 Common Relaxation Equations

This chapter discusses a GAMMA module that supplies commonly used relaxation equations. In most cases the equations were derived using a quantum mechanical treatment on a single spin or spin-pair that is dynamically moving as a randomly diffusing spherical top. In multiple spin systems the relaxation values returned by these functions employ a sum over spins / spin pairs.

There are three types of interactions accounted for herein. Another important to relaxation mechanism is due to the anisotropic part of the chemical shift Hamiltonian (CSA). Since the energies involved are dependent on the static field strength, this type of relaxation becomes more important as the field strength increases. Spins having spin angular momentum quantum values larger than 1/2, $I \geq 1$, may posses an appreciable electric quadrupole moment which provides an important relaxation mechanism.

3.1 Available Relaxation Functions

3.2 Covered Relaxation Theory

CSA

3.3 Relaxation Figures

3.4 Relaxation Example Programs

3.5 CSA Relaxation

3.5.1 R1_CC

Usage:

```
#include <relax_CSA.h>
row_vector R1_CC(sys_dynamic &dsys);
double R1_CC(sys_dynamic &dsys, int spin);
```

Description:

The function ***R1_CC*** returns a value(s) for the longitudinal relaxation rate expected from chemical shift anisotropy.

1. R1_CC(sys_dynamic &dsys) - The longitudinal relaxation rates of all spins in the system ***dsys*** are returned in a row vector.
2. double R1_CC(spin_sys &sys, int spin) - The longitudinal relaxation rate resulting from chemical shift anisotropy for ***spin*** of system ***dsys*** is returned.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***. Furthermore, it is assumed that the CSA tensor of each spin is symmetric.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <relax_CSA.h>
sys_dynamic dsys;                                  // Set up a dynamic system
dsys.read("filename.sys");                         // Read in system from file
cout << "\nThe CSA longitudinal relaxation rate of "
     << "spin 0 is " << R1_CC(sys, 0);             // Output R1 of spin 0
row_vector R1s = R1_CC(sys);                       // Get all R1's in a vector
```

Mathematical Basis:

For an isotropic spherical top and symmetric shift tensor, the expected CSA longitudinal relaxation rate is given by equations (2-2) and (2-3) on page 51.

$$R_1^{CSA} = \frac{1}{T_1^{CSA}} = \gamma^2 B_0^2 \Delta\sigma^2 \frac{2\tau}{15}\left[\frac{1}{1+(\omega\tau)^2}\right] = \frac{\xi^2\tau}{4\pi}\left[\frac{1}{1+(\omega\tau)^2}\right]$$

Here, $\Delta\sigma$ is the chemical shift anisotropy and ξ the GAMMA defined CSA interaction constant as given in Eq. (2-1) on page 49.

3.5.2 R1_CC_max

Usage:

```
#include <relax_CSA.h>
double R1_CC_max(sys_dynamic &dsys);
```

Description:

The function ***R1_CC_max*** compares the longitudinal CSA relaxation rates of all spin in the system ***dsys*** and returns the largest value. The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***. Furthermore, it is assumed that the CSA tensor of each spin is symmetric.

Return Value:

A double precision number is returned.

Examples:

```
#include <relax_CSA.h>
sys_dynamic dsys;                                   // Set up a dynamic system
dsys.read("filename.sys");                          // Read in system from file
cout << "\nThe fastest CSA longitudinal relaxation rate "
     << "predicted is " << R1_CC_max(dsys);         // Output R1 maximum
```

Mathematical Basis:

See the discussion for the function R1_CC on page 39.

3.5.3 R2_CC

Usage:

```
#include <relax_CSA.h>
row_vector R2_CC(sys_dynamic &dsys);
double R2_CC(sys_dynamic &dsys, int spin);
```

Description:

The function ***R2_CC*** returns a value(s) for the transverse relaxation rate expected from chemical shift anisotropy.

1. row_vector R2_CC(sys_dynamic &dsys) - The transverse relaxation rates of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double R2_CC(spin_sys &sys, int spin) - The transverse relaxation rate resulting from dipole-dipole interactions for ***spin*** of system ***dsys*** is returned based on a two-spin approximation.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***. Furthermore, it is assumed that the CSA tensor of each spin is symmetric.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <relax_CSA.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
row_vector R2s = R2_CC(dsys);              // Get all R2 values
double R20 = R2_CC(dsys, 0);               // Get R2 value of spin 0
```

Mathematical Basis:

For an isotropic spherical top and symmetric shift tensor, the expected CSA transverse relaxation rate is given by equations (2-6) and (2-7) on page 53.

$$R_2^{CSA} = \frac{1}{T_2^{CSA}} = \gamma^2 B_0^2 \Delta\sigma^2 \frac{\tau}{45}\left[\frac{3}{1+(\omega\tau)^2}+4\right] = \xi^2 \frac{\tau}{24\pi}\left[\frac{3}{1+(\omega\tau)^2}+4\right]$$

Here, $\Delta\sigma$ is the chemical shift anisotropy and ξ the GAMMA defined CSA interaction constant as given in the Eq. (2-1) on page 49.

3.5.4 R2_CC_max

Usage:

```
#include <relax_CSA.h>
double R2_CC_max(sys_dynamic &dsys);
```

Description:

The function ***R2_CC_max*** compares the transverse CSA relaxation rates of all spin in the system ***dsys*** and returns the largest value. The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***. Furthermore, it is assumed that the CSA tensor of each spin is symmetric.

Return Value:

A double precision number is returned.

Examples:

```
#include <relax_CSA.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
cout << "\nThe fastest CSA transverse relaxation rate "
     << "predicted is " << R2_CC_max(dsys);    // Output R2 maximum
```

Mathematical Basis:

See the discussion for the function R2_CC on page 40.

3.5.5 T1_CC

Usage:

```
#include <relax_CSA.h>
row_vector T1_CC(sys_dynamic &dsys);
double T1_CC(sys_dynamic &dsys, int spin);
```

Description:

The function ***T1_CC*** returns a value(s) for the longitudinal relaxation time expected from chemical shift anisotropy.

1. T1_CC(sys_dynamic &dsys) - The longitudinal relaxation times of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double T1_CC(spin_sys &sys, int spin) - The longitudinal relaxation time resulting from dipole-dipole interactions for ***spin*** of system ***dsys*** is returned based on a two-spin approximation.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***. Furthermore, it is assumed that the CSA tensor of each spin is symmetric.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <relax_CSA.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
cout << "\nThe T1 time for spin 0 is "
      << T1_CC(sys,0);                     // Output the T1 time of spin 0
row_vector T1s = T1_CC(sys);               // Get all the T1 times
cout << "\nThe T1 time for spin 1 is "
     << Re(T1s.get(0,1));                  // Output the T1 time of spin 1
```

Mathematical Basis:

See the discussion for the function R1_CC on page 39.

3.5.6 T1_CC_max

Usage:

```
#include <relax_CSA.h>
double T1_CC_max(sys_dynamic &dsys);
```

Description:

The function ***T1_CC_max*** compares the longitudinal CSA relaxation times of all spin in the system ***dsys*** and returns the largest value. The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***. Furthermore, it is assumed that the CSA tensor of each spin is symmetric.

Return Value:

A double precision number is returned.

Examples:

```
#include <relax_CSA.h>
sys_dynamic dsys;                                   // Set up a dynamic system
dsys.read("filename.sys");                          // Read in system from file
cout << "\nThe largest CSA longitudinal relaxation time "
     << "predicted is " << T1_CC_max(dsys);         // Output T1 maximum
```

Mathematical Basis:

See the discussion for the function R1_CC on page 39.

3.5.7 T2_CC

Usage:

```
#include <relax_CSA.h>
row_vector T2_CC(sys_dynamic &dsys);
double T2_CC(sys_dynamic &dsys, int spin);
```

Description:

The function ***T2_CC*** returns a value(s) for the transverse relaxation time expected from chemical shift anisotropy.

1. R2_CC(sys_dynamic &dsys) - The transverse relaxation times of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double R2_CC(spin_sys &sys, int spin) - The transverse relaxation time resulting from dipole-dipole interactions for ***spin*** of system ***dsys*** is returned based on a two-spin approximation.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***. Furthermore, it is assumed that the CSA tensor of each spin is symmetric.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <relax_CSA.h>
sys_dynamic dsys;                                   // Set up a dynamic system
dsys.read("filename.sys");                          // Read in system from file
row_vector T2s = T2_CC(dsys);                       // Get all the T2 times
int i = 1;                                          // Set a spin index variable
double T2i = T2_CC(dsys, i);                        // Set H in its eigenbasis
```

Mathematical Basis:

See the discussion for the function R2_CC on page 40.

3.5.8 T2_CC_max

Usage:

```
#include <relax_CSA.h>
double T2_CC_max(sys_dynamic &dsys);
```

Description:

The function ***T2_CC_max*** compares the transverse CSA relaxation times of all spin in the system ***dsys*** and returns the largest value. The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***. Furthermore, it is assumed that the CSA tensor of each spin is symmetric.

Return Value:

A double precision number is returned.

Examples:

```
#include <relax_CSA.h>
sys_dynamic dsys;                                   // Set up a dynamic system
dsys.read("filename.sys");                          // Read in system from file
cout << "\nThe largest CSA transverse relaxation time "
     << "predicted is " << T2_CC_max(dsys);         // Output T2 maximum
```

Mathematical Basis:

See the discussion for the function R2_CC on page 40.

3.5.9 LWhh_CC

Usage:

```
#include <relax_CSA.h>
row_vector LWhh_CC(sys_dynamic &dsys);
double LWhh_CC(sys_dynamic &dsys, int spin);
```

Description:

The function ***LWhh_CC*** returns a value(s) for the linewidths (at half-height) expected from chemical shift anisotropy.

1. LWhh_CC(sys_dynamic &dsys) - The linewidths of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double LWhh_CC(sys_dynamic &sys, int spin) - The linewidth resulting from dipole-dipole interactions for ***spin*** of system ***dsys*** is returned based on a two-spin approximation.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***. Furthermore, it is assumed that the CSA tensor of each spin is symmetric.

Return Value:

Either a row vector a double precision number is returned.

Examples:

```
#include <relax_CSA.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
double LW = LWhh_CC(dsys, 0);              // Set LW to CSA line width of spin 0
row_vector LWs = LWhh_CC(dsys);            // Get all CSA linewidths
```

Mathematical Basis:

The line-width at half-height is related to the transverse relaxation rate by the simple formula

$$LW_{hh}^{CSA} = R_2^{CSA}/\pi = 1/(\pi T_2^{CSA})$$

3.5.10 LWhh_CC_max

Usage:

```
#include <relax_CSA.h>
double LWhh_CC_max(sys_dynamic &dsys, int spin1);
```

Description:

The function ***LWhh_CC_max*** considers the expected linewidths due to CSA relaxation of all spin in the system ***dsys*** and returns the maximum value. The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***. Furthermore, it is assumed that the CSA tensor of each spin is symmetric.

Return Value:

A double precision number is returned.

Examples:

```
#include <relax_CSA.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
cout << "\nThe maximum linewidth expected "
     << "from CSA relaxation is "
     << LWhh_CC_max(dsys);                 // Output the max. CSA linewidth
```

Mathematical Basis:

The line-width at half-height is related to the transverse relaxation rate by the simple formula

$$LW_{hh}^{CSA} = R_2^{CSA}/\pi = 1/(\pi T_2^{CSA})$$

3.5.11 xiCSA

Usage:

```
#include <relax_CSA.h>
matrix xiCSA(sys_dynamic &dsys);
double xiCSA(sys_dynamic& sys, int i);
matrix xiCSA(spin_system &sys, double* CSAs);
double xiCSA(spin_system &sys, int i, double CSA);
```

Description:

This function ***xiCSA*** calculates the GAMMA defined CSA interaction constant as given in the CSA Hamiltonian presented in the Eq. (2-1) on page 49. It relates to the chemical shift anisotropy according to

$$\xi_i^{CSA} = \sqrt{\frac{8\pi}{15}}\gamma_i B_o \Delta\sigma_i$$

1. xiCSA(sys_dynamic &dsys) - The dynamic spin system ***dsys*** furnishes all components needed for the calculation over all spins in the system. A diagonal matrix whose elements contain the xi values for the system spins is returned.
2. xiCSA(sys_dynamic &dsys, int i) - As in the previous function, the dynamic spin system ***dsys*** furnishes all components needed for the calculation. In this case the interaction constant for the spin ***i*** is returned.
3. xiCSA(spin_system &sys, double *CSA) - This function is similar to the first form but obtains the field strength and gyromagnetic ratios from the input spin system ***sys***. The values of the chemical shift anisotropy $\Delta\sigma$ are input in the double precision vector and assumed to be in PPM.
4. xiCSA(spin_system &sys, int i, double CSA) - This function also obtains the field strength and gyromagnetic ratio of spin ***i*** from the input spin system ***sys***. The value of the chemical shift anisotropy $\Delta\sigma_i$ is input as a double precision number assumed to be in PPM.

Return Value:

Either a row vector or a double is returned.

Example(s):

```
#include <relax_CSA.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
matrix xis = xiCSA(dsys);                  // Matrix of CSA interaction consts.
double xi0 = xiCSA(dsys, 0);               // CSA interaction const. spin 0
sys_dynamic sys;                           // Set up a spin system
sys.read("filename.sys");                  // Read in system from file
double xi = xiCSA(sys, 0, 50);             // CSA int. const. spin 0 w/ 50 PPM
double csas[sys.spins()];                  // Set up vector of CSAs
for(int i=0; i<sys.spins(); i++)           // Fill up vector with CSA values
  csas[i] = 50.0 + i*10.0;
xis = xiCSA(sys, csas);                    // Reset matrix to new xi values
```

3.5.12 CSA

Usage:

```
#include <relax_CSA.h>
row_vector CSA(sys_dynamic &dsys);
double CSA(sys_dynamic& sys, int i);
```

Description:

This function ***CSA*** returns the chemical shift anisotropy value(s) for the spins in the system.

1. CSA(sys_dynamic &dsys) - The dynamic spin system ***dsys*** contains any spatial shielding tensors assigned to the spins in the system. A row vector containing the CSA values (PPM) of these tensors is returned.
2. CSA(sys_dynamic &dsys, int i) - The CSA value in PPM for the spin ***i*** of system ***dsys*** is returned.

Return Value:

Either a row vector or a double is returned.

Example(s):

```
#include <relax_CSA.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
cout << "\nThe CSA of spin 0 is "
     << CSA(dsys, 0) << " PPM";            // Output the CSA of spin 0
row_vector CSAs = CSA(dsys);               // Get all the system CSA values
```

Mathematical Basis:

The anisotropy of the shielding tensor is defined to be

$$\Delta\sigma = \sigma_{zz} - \frac{1}{2}(\sigma_{xx} + \sigma_{yy}) = \frac{3}{2}\delta_{zz} = \sigma_{\parallel} - \sigma_{\perp}$$

The last relationship uses the nomenclature applicable in describing a symmetric CSA tensor with $\eta = 0$, where $\sigma_{\parallel} = \sigma_{zz}$ and $\sigma_{\perp} = \sigma_{xx} = \sigma_{yy}$.

3.6 CSA Relaxation Discussion

For convenience the following lists the sections, figures, tables, and example GAMMA programs contained in this Chapter.

3.6.0.1 CSA Relaxation Sections

3.6.0.2 CSA Relaxation Figures

3.6.0.3 CSA Relaxation Tables

3.6.0.4 CSA Relaxation Example Programs

3.6.1 The CSA Interaction Constant

The CSA interaction constant, ξ_i^{CSA}, is used throughout GAMMA. It is simply a scaling factor which allows for independent scaling of spatial and spin tensors associated with the chemical shift anisotropy Hamiltonian (and others). Those interested in its origin must peruse the GAMMA documentation on the shift anisotropy interaction. Since this constant is implicit, rather than explicit, to the functions described in this chapter, we merely present what it is.

$$\xi_i^{CSA} = \sqrt{\frac{6\pi}{5}}\gamma_i B_o \delta_{zz}(i) \tag{2-1}$$

The CSA interaction constant in not of much consequence unless users wish to calculate related quantities. For their sake it will now be explicitly calculated. The first thing to note is that the factor $\gamma_i B_o$ is simply the Larmor frequency for isotope i, in the case of a proton this is the spectrometer frequency Ω_{spec}.

$$\xi_i^{CSA} = \sqrt{\frac{6\pi}{5}}\gamma_i B_o \delta_{zz}(i) = \sqrt{\frac{6\pi}{5}}\frac{\gamma_i}{\gamma_H}\Omega_{spec}\delta_{zz}(i) = \sqrt{\frac{6\pi}{5}}\frac{\gamma_i}{\gamma_H}2\pi\Omega_{spec,\nu}\delta_{zz}(i)$$

For a proton in a 500MHz spectrometer with a δ_{zz} value of 1 Part Per Million (PPM) we have

$$\xi_{^1H}^{CSA}\Big|_{\substack{500MHz\\1PPM}} = \sqrt{\frac{6\pi}{5}}\left(\frac{\gamma_{^1H}}{\gamma_{^1H}}\right)[2\pi(500\times10^6 Hz)](1/10^6)$$

$$= \sqrt{3.77}(1)[2\pi(500Hz)] = (1.942)(3.142\times10^3\text{sec}^{-1}) = 6.100\times10^3\text{sec}^{-1}$$

A more typical value would result from an anisotropy, $\Delta\sigma$, of 150 PPM for a carbon nucleus[1].

$$\xi_{^{13}C}^{CSA}\Big|_{\substack{500MHz\\150PPM}} = 2\pi\sqrt{\frac{6\pi}{5}}\left(\frac{\gamma_{^{13}C}}{\gamma_{^1H}}\right)500\times10^6 Hz(100/10^6) = 2\pi\sqrt{37.7}125.7Hz = 1.534\times10^5\text{sec}^{-1}$$

That the CSA interaction constant is linearly dependent on field strength[2] is shown below.

1. Note that $\Delta\sigma = 150$ implies that $\delta_{zz} = 100$ as discussed in the paragraph following the figure.

2. This plot was generated with the program Xi_CSA.cc given at the end of this Chapter, page 59.

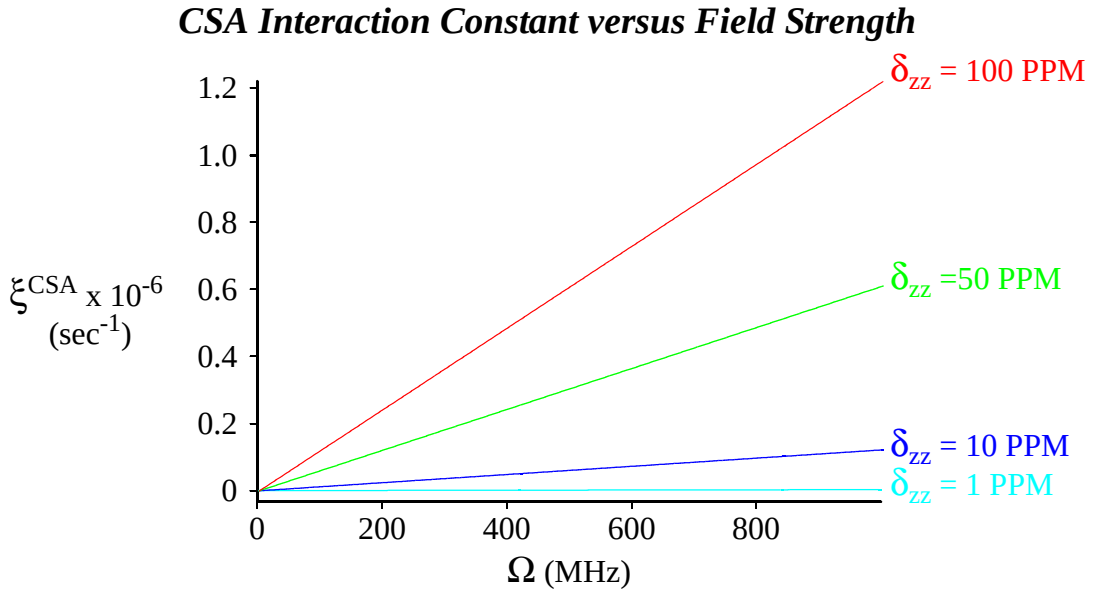


Figure 0-4 **CSA interaction constant relative to the applied field strength and the anisotropy.**

Keep in mind that the magnitude of ξ_i^{CSA} depends upon the field strength present. Pure CSA relaxation rates are proportional ξ^2 and thus proportional to the field strength squared. It is also of interest to note that the δ_{zz} value is not the chemical shift anisotropy (CSA), $\Delta\sigma$, although the two values are related. The following relationship is useful in discerning a proper δ_{zz} value to be used in GAMMA.

$$\Delta\sigma = \sigma_{zz} - \frac{1}{2}(\sigma_{xx} + \sigma_{yy}) = \frac{3}{2}\delta_{zz} = \sigma_{\parallel} - \sigma_{\perp}$$

The last relationship used the nomenclature applicable in describing a symmetric CSA tensor with $\eta = 0$, where $\sigma_{\parallel} = \sigma_{zz}$ and $\sigma_{\perp} = \sigma_{xx} = \sigma_{yy}$.

3.6.2 CSA Spin-Lattice Relaxation

We now consider the spin lattice or T_1 relaxation expected from the CSA of a spin. An equation commonly found in the literature is[1].

$$R_1^{CSA} = \frac{1}{T_1^{CSA}} = \gamma^2 B_0^2 (\sigma_{\parallel} - \sigma_{\perp})^2 \frac{2\tau}{15}\left[\frac{1}{1 + (\omega\tau)^2}\right] = \gamma^2 B_0^2 \Delta\sigma^2 \frac{2\tau}{15}\left[\frac{1}{1 + (\omega\tau)^2}\right] \qquad (2\text{-}2)$$

which applies to a spin having a symmetric shielding tenor - a tensor with $\eta = 0$ or equivalently with $\sigma_{\parallel} = \sigma_{zz}$ and $\sigma_{\perp} = \sigma_{yy} = \sigma_{xx}$. Also this equation is restricted to the dynamical case of a spherical top undergoing random rotational motion. Replacing $\Delta\sigma$ first with $(3/2)\delta_{zz}$ and subsequently putting the formula into GAMMA nomenclature using the CSA interaction constant

1. See *Pulse and Fourier Transform NMR* by Thomas C. Farrar and Edwin D. Becker, Academic Press, New York, New York, 1971. Specifically this equation is 4.28 on page 59. Also see "Calculation of Nuclear Spin Relaxation Times" by James L. Sudmeier, *et. al.*, *Conc. Magn. Reson.*, **1990**, 2, 197-212, specifically page 202, equation [35].

yields

$$R_1^{CSA} = \frac{1}{T_1^{CSA}} = \gamma^2 B_0^2 \delta_{zz}^2 \frac{3\tau}{10}\left[\frac{1}{1+(\omega\tau)^2}\right] = \frac{\xi^2\tau}{4\pi}\left[\frac{1}{1+(\omega\tau)^2}\right] \quad (2\text{-}3)$$

This longitudinal relaxation equation predicts how the correlation time affects T_1 times based on the chemical shielding anisotropy. The following figure was generated by a GAMMA program[1] for a proton in a field strength of 500 MHz. Keep in mind that this simple treatment assumes that the system containing the spin moves as a spherical top with isotropic motions. Furthermore note that for small values of τ where $\omega\tau \ll 1$ (the extreme narrowing condition), R_1^{CSA} increases and T_1^{CSA} decreases linearly with the correlations time: as the molecule begins to slow, τ increases, the relaxation rate increases, and the relaxation time becomes shorter. The opposite is true when we are far away from the extreme narrowing, when $\omega\tau \gg 1$. Then, as the molecule further slows down (heading toward a solid) CSA no longer provides a nice longitudinal relaxation pathway. It is then T_1^{CSA} which increases linearly with τ. It takes longer for longitudinal relaxation to occur.

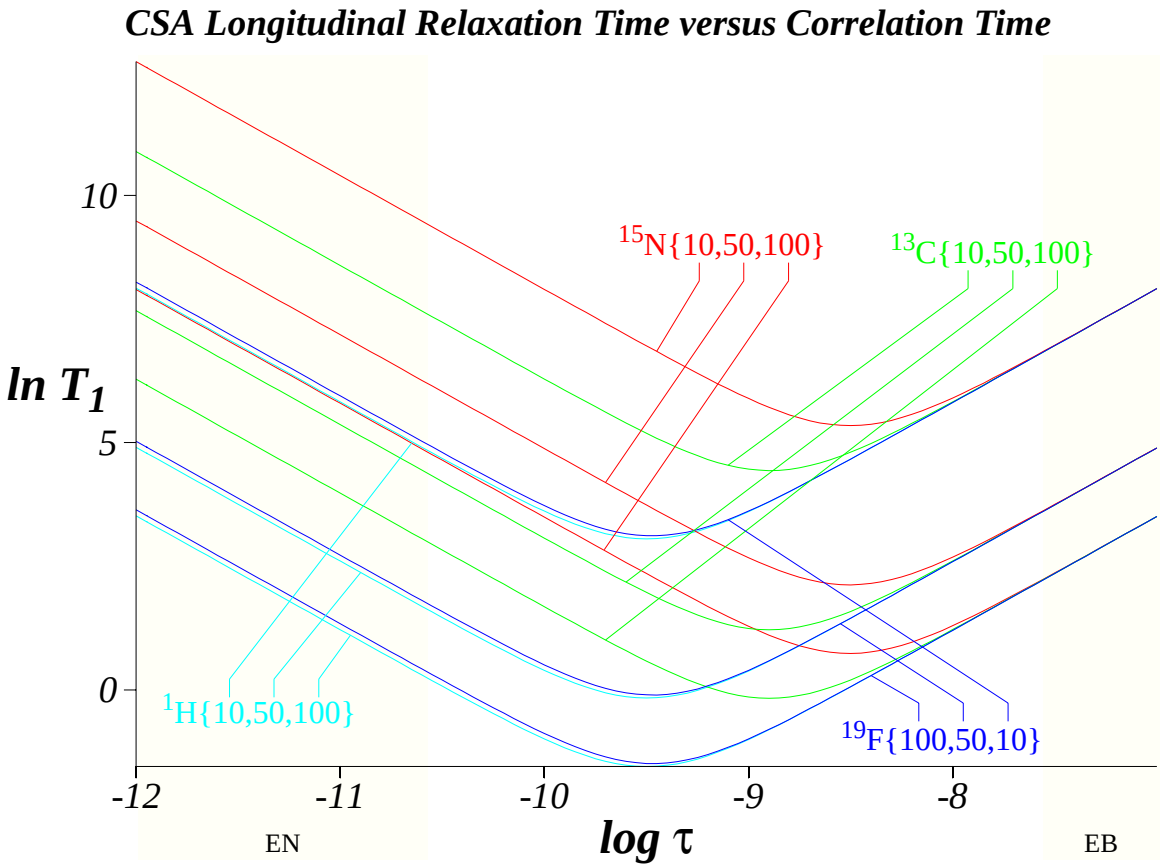


Figure 0-5 **Natural log of the CSA longitudinal relaxation time versus the base 10 log of the correlation time. The applied field strength was set to 500 MHz. The isotopes are noted with the δ_{zz} delz values indicated in brackets. The shaded portion to the left is the Extreme**

1. This figure was produced by the GAMMA program listed at the end of this chapter, T1plot_CSA.cc, page 59.

Narrowing (EN) region where $\omega\tau \ll 1$ and the relationship is linear. Linear behavior is also seen in the shaded area on the right (EB) where $\omega\tau \gg 1$.

We can estimate the spin lattice CSA relaxation rate under extreme narrowing (EN) conditions. Recall that for extreme narrowing $\omega\tau \ll 1$. In these instances, equation (2-3) becomes

$$R_1^{CSA}\Big|_{EN} = \frac{\xi^2\tau}{4\pi} \tag{2-4}$$

and it is apparent that the relaxation rate is proportional to the correlation time. Using the CSA interaction constant previously calculated for a ^{13}C nucleus with field strength of 500 MHz and a δ_{zz} of 100 PPM, $\xi^{CSA} = 1.534 \times 10^5 \text{sec}^{-1}$, and a correlation time of 1 picosecond we can directly calculate the spin lattice relaxation time expected in the extreme narrowing limit.

$$T_1^{CSA}\Big|_{EN} = \left[\frac{\xi^2\tau}{4\pi}\right]^{-1} = \left[\frac{(1.534 \times 10^5 \text{sec}^{-1})^2 1.0 \times 10^{-12}\text{sec}}{4\pi}\right]^{-1} = 534 \text{ sec} \approx 8.9 \; min$$

Notice that this corresponds to $ln(T_1^{CSA}) = 6.28$ for ^{13}C at $\delta_{zz} = 100$ which can be seen in the previous plot. The other end of the motional spectrum occurs when $\omega\tau \gg 1$ but at common field strengths where $\omega \sim 10^8$ we would need extremely slow motion; slower than most large proteins which are moving at $\tau \sim 10^{-8}$. Under such circumstances, the motion is not likely to be isotropic. We shall label this regime as EB, and

$$R_1^{CSA}\Big|_{EB} \not\propto \frac{\xi^2}{4\omega^2\pi}\left[\frac{1}{\tau}\right] \tag{2-5}$$

3.6.3 CSA Spin-Spin Relaxation

We now consider the spin-spin (transverse) or T_2 relaxation expected from the CSA of a spin. For this simple treatment there exists an equation typically found in the literature[1].

$$R_2^{CSA} = \frac{1}{T_2^{CSA}} = \gamma^2 B_0^2 (\sigma_{\parallel} - \sigma_{\perp})^2 \frac{\tau}{90}\left[\frac{6}{1 + (\omega\tau)^2} + 8\right] = \gamma^2 B_0^2 \Delta\sigma^2 \frac{\tau}{45}\left[\frac{3}{1 + (\omega\tau)^2} + 4\right] \tag{2-6}$$

Again, this applies to a spin having a symmetric shielding tenor - a tensor with $\eta = 0$ or equivalently with $\sigma_{\parallel} = \sigma_{zz}$ and $\sigma_{\perp} = \sigma_{yy} = \sigma_{xx}$ - it is restricted to the dynamical case of a spherical top undergoing random rotational motion. Replacing $\Delta\sigma$ with $(3/2)\delta_{zz}$ produces

$$R_2^{CSA} = \frac{1}{T_2^{CSA}} = \gamma^2 B_0^2 \delta_{zz}^2 \frac{\tau}{20}\left[\frac{3}{1 + (\omega\tau)^2} + 4\right]$$

1. Also found in the previous reference, *Pulse and Fourier Transform NMR* by Thomas C. Farrar and Edwin D. Becker, Academic Press, New York, New York, 1971. Specifically see equation are 4.29 on page 59. In the article "Calculation of Nuclear Spin Relaxation Times" by J.L. Sudmeier, S.E. Anderson, and J.S. Frye, *Conc. Magn. Reson.*, **1990**, 2, 197-212, this corresponds to page 203, equation [36].

and we now place the formula into full GAMMA nomenclature using the CSA interaction constant.

$$R_2^{CSA} = \frac{1}{T_2^{CSA}} = \xi^2 \frac{\tau}{24\pi}\left[\frac{3}{1+(\omega\tau)^2} + 4\right] \tag{2-7}$$

We now show graphically how T_2 varies with correlation time in accordance with equation (2-7). The figure below applies to a single spin system at 500 MHz[1].

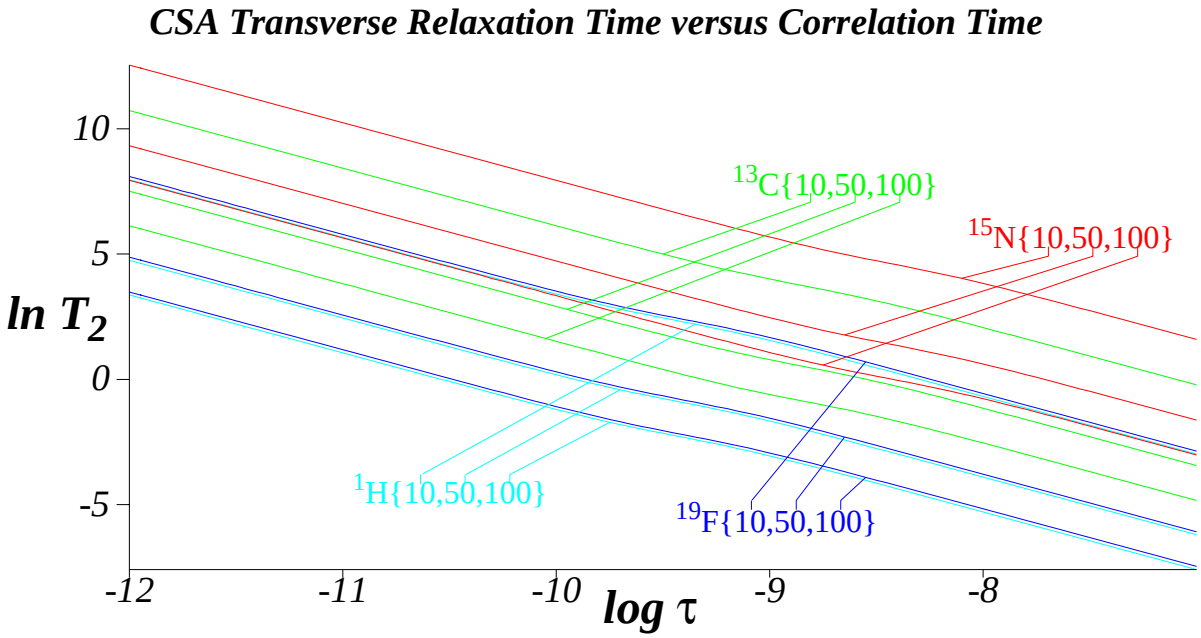


Figure 0-6 **Natural log of the CSA transverse relaxation time versus the base 10 log of the correlation time. The applied field strength was set to 500 MHz. The isotopes are noted with the δ_{zz} values indicated in brackets.**

Under extreme narrowing, $\omega\tau \ll 1$, the spin-spin CSA relaxation rate is

$$R_2^{CSA}\Big|_{EN} = \frac{7}{24\pi}\xi^2\tau \tag{2-8}$$

Using the CSA interaction constant previously calculated for a ^{13}C nucleus with field strength of 500 MHz and a δ_{zz} of 100 PPM, $\xi^{CSA} = 1.534 \times 10^5 \text{sec}^{-1}$, and a correlation time of 1 picosecond we can directly calculate the spin lattice relaxation time expected in the extreme narrowing limit.

$$T_2^{CSA}\Big|_{EN} = \left[\frac{7}{24\pi}\xi^2\tau\right]^{-1} = \left[\frac{(1.534\times10^5\text{sec}^{-1})^2 7.0\times10^{-12}\text{sec}}{24\pi}\right]^{-1} = 458 \text{ sec} \approx 7.6 \text{ min}$$

Notice that this corresponds to $ln(T_2^{CSA}) = 6.13$ for ^{13}C at $\delta_{zz} = 100$ which can be seen in the previous plot. If we compare the longitudinal and transverse relaxation times in the extreme nar-

1. This figure was produced by the GAMMA program listed at the end of this Chapter, T2plot_CSA.cc, page 61.

rowing limit, the inverses of equations (2-4) and (2-8), we see the well know ratio of 7/6.

$$\frac{T_1^{CSA}\Big|_{EN}}{T_2^{CSA}\Big|_{EN}} = \frac{R_2^{CSA}\Big|_{EN}}{R_1^{CSA}\Big|_{EN}} = \frac{\frac{7}{24\pi}\xi^2\tau}{\frac{\xi^2\tau}{4\pi}} = \frac{28\pi}{24\pi} = \frac{7}{6} \tag{2-9}$$

In the extreme narrowing limit, CSA transverse relaxation occurs slightly faster than longitudinal relaxation; the transverse relaxation time is a little shorter than the longitudinal relaxation time.

3.6.4 CSA Relaxation Linewidths

The linewidths expected from CSA relaxation may be estimated directly from the spin-spin relaxation times according to the following relationship.

$$LW_{hh}^{CSA} = R_2^{CSA}/\pi = 1/(\pi T_2^{CSA}) \tag{2-10}$$

Here LW_{hh}^{CSA} is used to indicate the CSA related line-width at half-height.

3.6.5 CSA Relaxation Equations

We now group together the important equations regarding a the simple treatment of CSA relaxation.

CSA Relaxation Equations

Interaction Constant

$$\xi_i^{CSA} = \sqrt{\frac{6\pi}{5}}\gamma_i B_o \delta_{zz}(i) = \sqrt{\frac{6\pi}{5}}\frac{\gamma_i}{\gamma_H}\Omega_{spec}\delta_{zz}(i) = \sqrt{\frac{6\pi}{5}}\frac{\gamma_i}{\gamma_H}2\pi\Omega_{spec,\nu}\delta_{zz}(i)$$

Longitudinal Relaxation (Spin-Lattice)

$$R_1^{CSA} = \frac{1}{T_1^{CSA}} = \frac{\xi^2\tau}{4\pi}\left[\frac{1}{1+(\omega\tau)^2}\right] \xrightarrow[\text{Narrowing}]{\text{Extreme}} \frac{\xi^2\tau}{4\pi}$$

Transverse Relaxation (Spin-Spin)

$$R_2^{CSA} = \frac{1}{T_2^{CSA}} = \xi^2\frac{\tau}{24\pi}\left[\frac{3}{1+(\omega\tau)^2}+4\right] \xrightarrow[\text{Narrowing}]{\text{Extreme}} \frac{7\xi^2\tau}{24\pi}$$

Linewidth at Half-Height

$$LW_{hh}^{CSA} = R_2^{CSA}/\pi = 1/(\pi T_2^{CSA})$$

Extreme Narrowing Ratio

$$T_1^{CSA}/T_2^{CSA}\big|_{EN} = \frac{7}{6}$$

Chemical Shift Anisotropy

$$\Delta\sigma = \sigma_{zz} - \frac{1}{2}(\sigma_{xx}+\sigma_{yy}) = \frac{3}{2}\delta_{zz} = \sigma_{\parallel} - \sigma_{\perp}$$

3.6.6 CSA Single Spin Relaxation

The following table[1] tabulates the CSA relaxation parameters and linewidths expected for a proton

1. This table was generated from the program T1T2_CSA.cc listed at the end of this Chapter, page 62. The output from the program was placed into this document as a table (in FrameMaker) by first placing the program output into a file then importing it as an ASCII file. The imported text is then converted into a Format B Table with the paragraphs treated as cells using 1 or more blank spaces as a cell. This new table is then unconverted (another Table option: convert to paragraphs) in a column by column fashion. This procedure allows a table ASCII output from the program as well as facile generation of the table in this text! It isn't as difficult as it sounds.

at a field strength of 500.12 MHz.

Table 3: Estimated Proton CSA Relaxation Times @ 500 MHz

δ_{zz} (PPM)	$\Delta\sigma$ (PPM)	tau (sec)	T_1 (sec)	T_2 (sec)	LW_{hh} (Hertz)
.666667	1	10^{-12}	759552	651041	4.88925e-07
		10^{-11}	76029.4	65131.3	4.8872e-06
		10^{-10}	8345.44	6771.17	4.70096e-05
		10^{-9}	8259.54	1065.81	.000299
		10^{-8}	75075.9	113.845	.002796
		10^{-7}	750008	11.3931	.027939
6.66667	10	10^{-12}	7595.52	6510.41	4.88925e-05
		10^{-11}	760.294	651.313	.000489
		10^{-10}	83.4544	67.7117	.004701
		10^{-9}	82.5954	10.6581	.029866
		10^{-8}	750.759	1.13845	.279599
		10^{-7}	7500.08	.113931	2.79389
33.3333	50	10^{-12}	303.821	260.416	.001222
		10^{-11}	30.4118	26.0525	.012218
		10^{-10}	3.33818	2.70847	.117524
		10^{-9}	3.30382	.426323	0.74664
		10^{-8}	30.0304	.045538	6.98997
		10^{-7}	300.003	.004557	69.8472
66.6667	100	10^{-12}	75.9552	65.1041	.004889
		10^{-11}	7.60294	6.51313	.048872
		10^{-10}	.834544	.677117	.470096
		10^{-9}	.825954	.106581	2.98656
		10^{-8}	7.50759	.011385	27.9599
		10^{-7}	75.0008	.001139	279.389

Normally, proton anisotropy values are very small. This is not true for other common nuclei in NMR. For comparison, the tables below contains the same calculated values for fluorine, carbon, and nitrogen atoms. The proton field strength is kept at 500 MHz although the appropriate Larmor

frequency for the particular isotope will be the value ultimately utilized in equations.

Table 4: Estimated Fluorine-19 and Carbon-13 CSA Relaxation Times @ 500 MHz

δ_{zz} (PPM)	$\Delta\sigma$ (PPM)	tau (sec)	T_1 (sec)	T_2 (sec)	LW_{hh} (Hertz)	T_1 (sec)	T_2 (sec)	LW_{hh} (Hertz)
	Fluorine 19					Carbon 13		
.666667	1	10^{-12}	857894	735334	4.32878e-07	1.20129e+07	1.02968e+07	3.09136e-08
		10^{-11}	85863.7	73560.7	4.32717e-06	1.20136e+06	1.0297e+06	3.09128e-07
		10^{-10}	9328.87	7615.72	4.17964e-05	120879	103242	3.08314e-06
		10^{-9}	8357.89	1194.85	.000266	19512.9	12327.4	2.58213e-05
		10^{-8}	75085.8	128.573	.002476	76201.3	1780.88	.000179
		10^{-7}	750009	12.8682	.024736	750120	180.171	.001767
6.66667	10	10^{-12}	8578.94	7353.34	4.32878e-05	120129	102968	3.09136e-06
		10^{-11}	858.637	735.607	.000433	12013.6	10297.0	3.09128e-05
		10^{-10}	93.2887	76.1572	0.00418	1208.79	1032.42	.000308
		10^{-9}	83.5789	11.9485	0.02664	195.129	123.274	.002582
		10^{-8}	750.858	1.28573	.247572	762.013	17.8088	.017874
		10^{-7}	7500.09	.128682	2.47362	7501.20	1.80171	0.17667
33.3333	50	10^{-12}	343.158	294.134	.001082	4805.15	4118.70	7.7284e-05
		10^{-11}	34.3455	29.4243	.010818	480.545	411.881	.000773
		10^{-10}	3.73155	3.04629	.104491	48.3515	41.2968	.007708
		10^{-9}	3.34315	.477939	.666005	7.80515	4.93096	.064553
		10^{-8}	30.0343	.051429	6.18929	30.4805	0.71235	.446845
		10^{-7}	300.003	.005147	61.8404	300.048	.072069	4.41676
66.6667	100	10^{-12}	85.7894	73.5334	.004329	1201.29	1029.68	.000309
		10^{-11}	8.58637	7.35607	.043272	120.136	102.970	.003091
		10^{-10}	.932887	.761572	.417964	12.0879	10.3242	.030831
		10^{-9}	.835789	.119485	2.66402	1.95129	1.23274	.258213
		10^{-8}	7.50858	.012857	24.7572	7.62013	.178088	1.78738
		10^{-7}	75.0009	.001287	247.362	75.0120	.018017	17.6670

Table 5: Estimated Nitrogen-15 CSA Relaxation Times @ 500 MHz

δ_{zz} (PPM)	$\Delta\sigma$ (PPM)	tau (sec)	T_1 (sec)	T_2 (sec)	LW_{hh} (Hertz)
33.3333	50	10^{-12}	29566.4	25342.6	1.25603e-05
		10^{-11}	2956.67	2534.27	.000126
		10^{-10}	295.964	253.536	.001255
		10^{-9}	32.5664	26.3842	.012064
		10^{-8}	32.9566	4.15536	.076602
			300.296	.443168	0.71826

Table 5: Estimated Nitrogen-15 CSA Relaxation Times @ 500 MHz

δ_{zz} (PPM)	$\Delta\sigma$ (PPM)	tau (sec)	T_1 (sec)	T_2 (sec)	LW_{hh} (Hertz)
66.6667	100	10^{-12}	7391.59	6335.65	5.02411e-05
		10^{-11}	739.167	633.568	.000502
		10^{-10}	73.9909	63.3840	.005022
		10^{-9}	8.14159	6.59606	.048258
		10^{-8}	8.23916	1.03884	.306409
		10^{-7}	75.0739	.110792	2.87304

3.7 Shift Anisotropy Source Codes

Xi_CSA.cc

Generate Plots of CSA Interaction Constants

```
/** Xi_CSA.cc ************************************************** c++          **
**                                                                            **
**                                                                            **
GAMMA Example Program                                                         **
**                                                                            **
** This program loops through spins (1H) of differing chemical shift         **
** anisotropies and examines their corresponding GAMMA defined CSA           **
** interaction constants, xi, versus field strength. Plots are output        **
** to a single file called xiCSAplot.mif produced in FrameMaker .mif         **
** format. Each plot contains 301 points where the spectrometer              **
** frequency spans [0, 1000] MHz.                                            **
**                                                                            **
********************************************************************************/

#include <relax_CSA.h>

main ()
 {
 int ns=4;                                          // Number of spins
 sys_dynamic sys(ns);                               // Construct a spin system
 sys.delz(0, 1);                                    // Set the delz values for each
 sys.delz(1, 10);                                   // and this is kept in PPM
 sys.delz(2, 50);
 sys.delz(3, 100);
 int npts = 300;                                    // Set the # points per plot
 row_vector plot(npts), plots[ns];                  // Storage for plots
 double Ominc = 1.0e3/double(npts-1);               // Increment Omega to 1000 MHz
 double xi;                                         // For the Xi CSA value
 for(int spin=0; spin<ns; spin++)
  {
  double Om = 0.0;                                  // Initial Omega at 0 MHz
  for(int i=0; i<npts; i++)
    {
    sys.Omega(Om);                                  // Set the field strength
    xi = sys.xiCSA(spin);                           // Get Xi for CSA spin
    plot.put(xi, i);                                // Store Xi CSA
    Om += Ominc;                                    // Increment the field strength
    }
  plots[spin] = plot;
  }
 FM_1Dm("xiCSAplot.mif", ns, plots, 19.0, 14.0, 0.0, 1000.0);
}
```

T1plot_CSA.cc

T1plot_CSA - Generate Plot of CSA T1 versus tau

```
/* T1plot_CSA.cc ************************************************-*-c++-          *-
**                                                                            **
**                                                                            **
Example program for the GAMMA Library                                         **
**                                                                            **
** This program constructs a plot of T1 versus tau for a single              **
** spin system under the effects of Chemical Shift Anisotropy                **
** (CSA) relaxation. The calculations are performed using a                  **
** simple analytic formula for the T1 time which assumes that                **
** spin motion is that of a spherical top under rotationally                 **
** diffusive motion. It is also assumed that the shift tensor                **
** is axially symmetric.                                                     **
**                                                                            **
********************************************************************************/

#include <relax_CSA.h>

main (int argc, char* argv[])
 {
 cout << "\n\n\t\t\tGAMMA NMR Checking Program";
 cout << "\n\t\t CSA T1 Relaxation - Single Spin System\n\n";

 sys_dynamic dsys(1);                               // Set up a 1 spin system
 dsys.Omega(500.0);                                 // Set Omega to 500 MHz.
 int npts = 101;                                    // Use 101 points each T1 vs. tau
 String types[4];                                   // Look at 4 isotope types
 types[0] = "1H";
 types[1] = "13C";
 types[2] = "15N";
 types[3] = "19F";
 double delzz[3];                                   // Set up 3 delzz values
 delzz[0] = 10;
 delzz[1] = 50;
 delzz[2] = 100;

 row_vector plot(npts), plots[16];                  // Storage for plots
 double taui = 1.e-12;                              // Start at 10**-12 correlation
 double lntauinc = 5.0/double(npts-1);              // Increment tau 10**5 sec
 double lntau, tau, T1;

 int k=0;
 for(int iso=0; iso<4; iso++)                       // Loop through all isotopes
  {
  dsys.isotope(0, types[iso]);                      // Set spin isotope type
  for(int dz=0; dz<3; dz++)
    {
```

```
    dsys.delz(0, delzz[dz]);                    // Set system delzz
    tau = 1.0e-12;                              // Start at 1 psec tau
    lntau = -12.0;
    for(int i=0; i<npts; i++)                   // Loop through all points
     {
     dsys.taux(tau);                            // Set tau value of system
     T1 = T1_CC(dsys, 0);                       // Calculate new T1 value
     if(i == 0)                                 // Output initial & final values
       {                                        // to screen so plots discernable
       cout << "\n" << types[iso]
            << " - " << dsys.delz(0);
       cout << ": initial ln(T1) = " << log(T1);
       }
     else if(i == npts-1)
       cout << ", final ln(T1) = " << log(T1);
     plot.put(log(T1), i);                      // Store new T1 value
     lntau += lntauinc;                         // Increment log of tau
     tau = pow(10.0, lntau);                    // Determine next tau
     }
    plots[k] = plot;                            // Store this T1 vs. tau plot
    k++;
    }
  }
 FM_1Dm("T1Cplot.mif",k,plots,19,14,-12,-7);    // Output all plots to FrameMaker
 cout << "\n\n";                                // Keep screen nice
}
```

T1plot_CSA.cc

T2plot_CSA - Generate Plot of CSA T2 versus tau

```
/* T2plot_CSA.cc ***********************************************-*-c++-          *-
**                                                                              **
**
Example program for the GAMMA Library                                           **
**                                                                              **
** This program constructs a plot of T2 versus tau for a single                 **
** spin system under the effects of Chemical Shift Anisotropy                   **
** (CSA) relaxation. The calculations are performed using a                     **
** simple analytic formula for the T2 time which assumes that                   **
** spin motion is that of a spherical top under rotationally                    **
** diffusive motion. It is also assumed that the shift tensor                   **
** is axially symmetric.                                                        **
**                                                                              **
*********************************************************************          */

#include <relax_CSA.h>

main (int argc, char* argv[])
 {
 cout << "\n\n\t\t\tGAMMA NMR Checking Program";
 cout << "\n\t\t CSA T2 Relaxation - Single Spin System\n\n";

 sys_dynamic dsys(1);                          // Set up a 1 spin system
 dsys.Omega(500.0);                            // Set Omega to 500 MHz.
 int npts = 101;                               // Use 101 points each T2 vs. tau
 String types[4];                              // Look at 4 isotope types
 types[0] = "1H";
 types[1] = "13C";
 types[2] = "15N";
 types[3] = "19F";
 double delzz[3];                              // Set up 3 delzz values
 delzz[0] = 10;
 delzz[1] = 50;
 delzz[2] = 100;

 row_vector plot(npts), plots[16];             // Storage for plots
 double taui = 1.e-12;                         // Start at 10**-12 correlation
 double lntauinc = 5.0/double(npts-1);         // Increment tau 10**5 sec
 double lntau, tau, T2;

 int k=0;
 for(int iso=0; iso<4; iso++)                  // Loop through all isotopes
 {
   dsys.isotope(0, types[iso]);                // Set spin isotope type
   for(int dz=0; dz<3; dz++)
     {
     dsys.delz(0, delzz[dz]);                  // Set system delzz
     tau = 1.0e-12;                            // Start at 1 psec tau
     lntau = -12.0;
     for(int i=0; i<npts; i++)                 // Loop through all points
       {
       dsys.taux(tau);                         // Set tau value of system
       T2 = 1.0/T2_CC(dsys, 0);                // Calculate new T2 value
       if(i == 0)                              // Output initial & final values
         {                                     // to screen so plots discernable
         cout << "\n" << types[iso] << " - " << dsys.delz(0);
         cout << ": initial ln(T2) = " << log(T2);
         }
       else if(i == npts-1)
         cout << ", final ln(T2) = " << log(T2);
       plot.put(log(T2), i);                   // Store new T2 value
       lntau += lntauinc;                      // Increment log of tau
       tau = pow(10.0, lntau);                 // Determine next tau
       }
     plots[k] = plot;                          // Store this T2 vs. tau plot
     k++;
     }
   }
 FM_1Dm("T2Cplot.mif",k,plots,19,14,-12,-7);   // Output all plots to FrameMaker
 cout << "\n\n";                               // Keep screen nice
 }
```

T1T2_CSA.cc

Generate Table of CSA Relaxation Values

```
/* T1T2_CSA.cc ************************************************-*-c++-  *-
**                                                                     **
**                                                                     **
Example program for the GAMMA Library                                  **
**                                                                     **
** This program performs a simple calculation of the T1, T2,           **
**                                                                     **
** linewidths associated with a single spin having CSA. The            **
** calculations are performed using a simple treatment and only        **
** chemical shift anisotropy interactions is considered. The           **
** analytical formulas used assume the spin system motion is           **
** that of a spherical top under rotationally diffusive motion.        **
** They also assume an axially symmetric chemical shift tensor.        **
**                                                                     **
*************************************************************************/

#include <relax_CSA.h>

main (int argc, char* argv[])
 {
 cout << "\n\n\t\t\tGAMMA NMR Checking Program";
 cout << "\n\t\t CSA Relaxation - Single Spin System\n\n";

 sys_dynamic dsys(1);                          // Set up a 1 spin system
 double bigO;                                  // Set the spectrometer frequency
 query_parameter(argc, argv, 1,
   "Spectrometer Frequency (MHz)? ", bigO);
 dsys.Omega(bigO);
 String iso;                                   // Set the isotope type
 query_parameter(argc, argv, 2,
          "Spin Isotope Type? ", iso);
 dsys.isotope(0, iso);
 double DELzz[4];                              // Set up 4 DELzz values
 DELzz[0] = 1;
 DELzz[1] = 10;
 DELzz[2] = 50;
 DELzz[3] = 100;
 double tau, delzz;                            // Variables for tau and delzz
 double taui = 1.e-12;                         // Start at 10**-12 correlation
 double R1,R2,T1,T2,LW;
 cout << "\n\n\tCSA Relaxation of "
     << dsys.symbol(0) << "\n";
 cout << "\ndelzz\tDELzz\ttau"
     << "\tT1\tT2\tLWHH";
 for(int i=0; i<4; i++)                        // Loop over the 4 DELzz values
   {
   delzz = 2.0*DELzz[i]/3.0;                   // Determine delzz
   dsys.delz(0, delzz);                        // Set system delzz
   dsys.taux(taui);                            // Set initial correlation time
   tau = dsys.taux();
   for(int j=-12; j<-6; j++)                   // Loop over different taus
     {
     R1 = R1_CC(dsys, 0);                      // Calculate R1
     R2 = R2_CC(dsys, 0);                      // Calculate R2
     T1 = 1.0/R1;                              // Calculate T1 from R1
     T2 = 1.0/R2;                              // Calculate T2 from R2
     LW = R2/PI;                               // Calculate Linewidth from R2
     cout << "\n" << delzz                     // Output in form which can be
     << "\t" << DELzz[i] << "\t" << tau        // placed into a FrameMaker table
     << "\t" << T1<< "\t" << T2 << "\t"
     << LW;
     tau *= 10.0;                              // Increase tau by 10x
     dsys.taux(tau);
     }
   }
 cout << "\n\n";                               // Tidy up output
 }
```

4 Quadrupolar Relaxation Equations

This chapter discusses a GAMMA module that supplies commonly used quadrupolar relaxation equations. In most cases the equations were derived using a quantum mechanical treatment on a single spin that is dynamically moving as a randomly diffusing spherical top. Quadrupolar relaxation applies only to spins having spin angular momentum quantum values larger than 1/2, $I \geq 1$. These spins may posses an appreciable electric quadrupole moment which provides an important relaxation mechanism.

4.1 Available Quadrupolar Relaxation Functions

4.2 Covered Quadrupolar Relaxation Theory

4.3 Quadrupolar Relaxation Figures

4.4 Quadrupolar Relaxation Example Programs

4.5 Quadrupolar Relaxation

4.5.1 R1_QQ

Usage:

```
#include <gamma.h>
row_vector R1_QQ(sys_dynamic &dsys);
double R1_QQ(sys_dynamic &dsys, int spin1);
```

Description:

The function ***R1_QQ*** returns a value(s) for the longitudinal relaxation rate expected from quadrupolar relaxation.

1. R1_QQ(sys_dynamic &dsys) - The longitudinal relaxation rates of all spins in the system ***dsys*** are returned in a row vector.
2. double R1_QQ(spin_sys &sys, int spin) - The longitudinal relaxation rate resulting from electric quadrupole moment for spin ***spin*** of system ***dsys*** is returned.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <relax_Quad.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
row_vector R1s = R1_QQ(sys);               // Vector of relaxation rates
double R10 = R1_QQ(sys, 0);                // Relaxation rate of spin 1
```

Mathematical Basis:

For an isotropic spherical top the expected quadrupolar longitudinal relaxation rate is given below.

$$R_1^Q = \frac{1}{T_1^Q} = (2I+3)(2I-1)(\xi^Q)^2\frac{1}{20}\left[1+\frac{\eta^2}{3}\right]\left[\frac{1}{1+(\omega\tau)^2}+\frac{4}{1+(2\omega\tau)^2}\right]$$

$$= \left[\frac{3\tau(2I+3)}{400I^2(2I-1)}\right]QCC^2\left[1+\frac{\eta^2}{3}\right]\left[\frac{2}{1+(\omega\tau)^2}+\frac{8}{1+(2\omega\tau)^2}\right]$$

4.5.2 R2_QQ

Usage:

```
#include <gamma.h>
row_vector R2_QQ(sys_dynamic &dsys);
double R2_QQ(sys_dynamic &dsys, int spin1);
```

Description:

The function ***R2_QQ*** returns a value(s) for the transverse relaxation rate expected from chemical shift anisotropy.

1. R2_QQ(sys_dynamic &dsys) - The transverse relaxation rates of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double R2_QQ(spin_sys &sys, int spin) - The transverse relaxation rate resulting from quadrupolar interactions for spin ***spin*** of system ***dsys*** is returned based on a two-spin approximation.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <relax_Quad.h>
sys_dynamic dsys;                       // Set up a dynamic system
dsys.read("filename.sys");              // Read in system from file
row_vector R2s = R2_QQ(sys);            // Vector of relaxation rates
double R20 = R2_QQ(sys, 0);             // Relaxation rate of spin 1
```

Mathematical Basis:

For an isotropic spherical top the expected quadrupolar transverse relaxation rate is.

$$R_2^Q = \frac{1}{T_2^Q} = (2I+3)(2I-1)(\xi^Q)^2\frac{1}{20}\left[1+\frac{\eta^2}{3}\right]\left[3+\frac{5}{1+(\omega\tau)^2}+\frac{2}{1+(2\omega\tau)^2}\right]$$

$$= \left[\frac{3\tau(2I+3)}{400I^2(2I-1)}\right]QCC^2\left[1+\frac{\eta^2}{3}\right]\left[3+\frac{5}{1+(\omega\tau)^2}+\frac{2}{1+(2\omega\tau)^2}\right]$$

4.5.3 T1_QQ

Usage:

```
#include <gamma.h>
row_vector T1_QQ(sys_dynamic &dsys);
double T1_QQ(sys_dynamic &dsys, int spin1);
```

Description:

The function ***T1_QQ*** returns a value(s) for the longitudinal relaxation time expected from chemical shift anisotropy.

1. T1_QQ(sys_dynamic &dsys) - The longitudinal relaxation times of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double T1_QQ(spin_sys &sys, int spin) - The longitudinal relaxation time resulting from quadrupolar

interactions for spin ***spin*** of system ***dsys*** is returned based on a two-spin approximation.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <relax_Quad.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
row_vector T1s = T1_QQ(sys);               // Vector of relaxation times
double T10 = T1_QQ(sys, 0);                // Relaxation time of spin 1
```

Mathematical Basis:

For an isotropic spherical top and symmetric shift tensor, the expected quadrupolar longitudinal relaxation time is

$$R_1^Q = \frac{1}{T_1^Q} = (2I+3)(2I-1)(\xi^Q)^2\frac{1}{20}\left[1+\frac{\eta^2}{3}\right]\left[\frac{1}{1+(\omega\tau)^2}+\frac{4}{1+(2\omega\tau)^2}\right]$$

$$= \left[\frac{3\tau(2I+3)}{400I^2(2I-1)}\right]QCC^2\left[1+\frac{\eta^2}{3}\right]\left[\frac{2}{1+(\omega\tau)^2}+\frac{8}{1+(2\omega\tau)^2}\right]$$

4.5.4 T2_QQ

Usage:

```
#include <gamma.h>
row_vector T2_QQ(sys_dynamic &dsys);
double T2_QQ(sys_dynamic &dsys, int spin1);
```

Description:

The function ***T2_QQ***returns a value(s) for the transverse relaxation time expected from chemical shift anisotropy.

1. R2_QQ(sys_dynamic &dsys) - The transverse relaxation times of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double R2_QQ(spin_sys &sys, int spin) - The transverse relaxation time resulting from quadrupolar interactions for spin ***spin*** of system ***dsys*** is returned based on a two-spin approximation.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <relax_Quad.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
row_vector T2s = T2_QQ(sys);               // Vector of relaxation times
double T20 = T2_QQ(sys, 0);                // Relaxation time of spin 1
```

Mathematical Basis:

For an isotropic spherical top and symmetric shift tensor, the expected quadrupolar transverse relaxation time is

$$R_2^Q = \frac{1}{T_2^Q} = (2I+3)(2I-1)(\xi^Q)^2\frac{1}{20}\left[1+\frac{\eta^2}{3}\right]\left[3+\frac{5}{1+(\omega\tau)^2}+\frac{2}{1+(2\omega\tau)^2}\right]$$

$$= \left[\frac{3\tau(2I+3)}{400I^2(2I-1)}\right]QCC^2\left[1+\frac{\eta^2}{3}\right]\left[3+\frac{5}{1+(\omega\tau)^2}+\frac{2}{1+(2\omega\tau)^2}\right]$$

4.5.5 LWhh_QQ

Usage:

```
#include <gamma.h>
row_vector LWhh_QQ(sys_dynamic &dsys);
double LWhh_QQ(sys_dynamic &dsys, int spin1);
```

Description:

The function ***LWhh_QQ*** returns a value(s) for the linewidths (at half-height) expected from chemical shift anisotropy.

1. LWhh_QQ(sys_dynamic &dsys) - The linewidths of all spins in the system ***dsys*** are returned in a row vector. Each spin is assumed interacting will all other spins in the system and a two-spin approximation is used.
2. double LWhh_QQ(spin_sys &sys, int spin) - The linewidth resulting from quadrupolar interactions for spin ***spin*** of system ***dsys*** is returned based on a two-spin approximation.

The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***.

Return Value:

Either a row vector a a double precision number is returned.

Examples:

```
#include <relax_Quad.h>
sys_dynamic dsys;                          // Set up a dynamic system
dsys.read("filename.sys");                 // Read in system from file
row_vector LWs = LWhh_QQ(sys);             // Vector of quadrupolar linewidths
double LW0 = LWhh_QQ(sys, 0);              // Linewidth of spin 1
```

Mathematical Basis:

The line-width at half-height is related to the transverse relaxation rate by the simple formula

$$LW_{hh}^{Q} = R_2^Q/\pi = 1/(\pi T_2^Q)$$

4.5.6 LWhh_QQ_max

Usage:

```
#include <gamma.h>
double LWhh_QQ_max(sys_dynamic &dsys);
```

Description:

The function ***LWhh_QQ_max*** returns a linewidth for the spin is system ***dsys*** which is relaxing the most rapidly due to quadrupolar effects. The computation assumes that the system moves as an isotropic manner characterized by a single correlation time. This is taken to be the first value listed ***dsys***.

Return Value:

A double precision number is returned.

Examples:

```
#include <relax_Quad.h>
sys_dynamic dsys;                       // Set up a dynamic system
dsys.read("filename.sys");              // Read in system from file
cout << "\nMax Linewidth: "
     << LWhh_QQ_max(sys);               // Output the max linewidth
```

Mathematical Basis:

The line-width at half-height is related to the transverse relaxation rate by the simple formula

$$LW_{hh}^{Q} = R_2^Q/\pi = 1/(\pi T_2^Q)$$

4.5.7 xiQ

Usage:

```
#include <gamma.h>
row_vector xiQ(sys_dynamic &dsys);
double xiQ(spin_system& sys, int i);
```

Description:

This function ***xiQ*** calculates the quadrupolar interaction constant according to

$$\xi_i^Q = \sqrt{\frac{6\pi}{5}}\frac{e^2 q_i Q_i}{2I_i(2I_i-1)h} = \sqrt{\frac{6\pi}{5}}\frac{QCC_i}{2I_i(2I_i-1)}$$

1. xiQ(sys_dynamic &dsys) - The dynamic spin system ***dsys*** furnishes all components needed for the cal-

culation over all spins in the system. A vector contains the xi values for each spin is returned.

2. xiQ(spin_sys &sys, int i) - As in the previous function, the dynamic spin system ***dsys*** furnishes all components needed for the calculation. In this case the interaction constant for the spin ***i*** is returned.

Return Value:

Either a row vector or a double is returned.

Example:

```
#include <gamma.h>
sys_dynamic dsys;                       // Set up a dynamic system
dsys.read("filename.dsys");             // Read in system from file
row_vector Xis = xiQ(dsys);             // Get all the system quad. xi values
double Xi0 = xiQ(dsys, 0);              // Get 1st spins quad xi value
```

4.5.8 QCC

Usage:

```
#include <gamma.h>
row_vector QCC(sys_dynamic &dsys);
double QCC(spin_system& sys, int i);
```

Description:

This function ***QCC*** provides access to spin system quadrupolar coupling constants according to

$$QCC_i = e^2 q_i Q_i$$

1. QCC(sys_dynamic &dsys) - The dynamic spin system ***dsys*** stores and maintains these coupling constants. A vector containing the coupling constant values is returned.
2. QCC(spin_sys &sys, int i) - As in the previous function, the dynamic spin system ***dsys*** furnishes all coupling constant values. In this case the constant for the spin ***i*** is returned.

Return Value:

Either a row vector or a double is returned.

Example:

```
#include <gamma.h>
sys_dynamic dsys;                       // Set up a dynamic system
dsys.read("filename.dsys");             // Read in system from file
row_vector Qs = QCC(dsys);              // Get system quad. coupling values
double Q0 = QCC(dsys, 0);               // Get 1st spins quad coupling value
```

4.6 Quadrupolar Relaxation Discussion

For convenience the following lists the sections, figures, tables, and example GAMMA programs contained in this Chapter.

4.6.0.1 Quadrupolar Relaxation Sections

4.6.0.2 Quadrupolar Relaxation Figures

4.6.0.3 Quadrupolar Relaxation Tables

4.6.0.4 Quadrupolar Relaxation Example Programs

4.6.1 The Quadrupolar Interaction Constant

The quadrupolar interaction constant, ξ_i^Q, is used throughout GAMMA. It is simply a scaling factor which allows for independent scaling of spatial and spin tensors associated with the quadrupolar Hamiltonian (and others). Those interested in its origin must peruse the GAMMA documentation on the quadrupolar interaction. Since this constant is implicit, rather than explicit, to the functions described in this chapter, we merely present what it is.

$$\xi_i^Q = \sqrt{\frac{6\pi}{5}}\frac{QCC_i}{2I_i(2I_i-1)} \tag{3-1}$$

The quadrupolar interaction constant in not of much consequence unless users wish to calculate related quantities. For their sake it will now be explicitly calculated for several species. We first consider the relaxation of deuterium.

$$\xi_D^Q = \frac{QCC_i}{2I_i(2I_i-1)}\sqrt{\frac{6\pi}{5}}\bigg|_{^2H} = \frac{QCC_D}{2(2-1)}\sqrt{\frac{6\pi}{5}} = (0.971)QCC_D$$

The aromatic deuterons on benzene-d_6 and the methyl deuterons on toluene-d_3 which have quadrupolar coupling constants of 193 and 165 kHz respectively[1].

$$\xi_D^Q\big|_{\varphi\text{-D}} = (0.971)2\pi(193\times10^3 Hz) = 1.177\times10^6\text{sec}^{-1}$$

$$\xi_D^Q\big|_{\varphi\text{-CD}_3} = (0.971)2\pi(165\times10^3 Hz) = 1.006\times10^6\text{sec}^{-1}$$

Another example would be value would be ^{35}Cl in trimethyl tin chloride which as a quadrupolar coupling constant of 29.0 MHz. The corresponding quadrupolar interaction constant is then[2].

$$\xi_{^{35}Cl}^Q = \frac{QCC_i}{2I_i(2I_i-1)}\sqrt{\frac{6\pi}{5}}\bigg|_{^{35}Cl} = \frac{2\pi(29\times10^6 Hz)}{2(3/2)[2(3/2)-1]}\sqrt{\frac{6\pi}{5}} = \frac{\pi(29\times10^6 Hz)}{3}\sqrt{\frac{6\pi}{5}}$$

$$\xi_{^{35}Cl}^Q = \frac{\pi(29\times10^6 Hz)}{3}(1.942) = 2.033(29\times10^6 Hz) = 58.96\times10^6\text{sec}^{-1}$$

4.6.2 Quadrupolar Spin-Lattice Relaxation

We now consider the spin lattice or T_1 relaxation expected from the electric quadrupole of a spin. An equation commonly found in the literature is

$$R_1^Q = \frac{1}{T_1^Q} = \left[\frac{3\tau(2I+3)}{400I^2(2I-1)}\right]\left[\frac{e^2Qq}{h}\right]^2\left[1+\frac{\eta^2}{3}\right]\left[\frac{2}{1+(\omega\tau)^2}+\frac{8}{1+(2\omega\tau)^2}\right]$$

1. Taken from Ando, Gerig, and Weigand, JACS, 104, 11, (1982) 3172-3178.
2. See “Calculation of Nuclear Spin Relaxation Times” by James L. Sudmeier, *et. al.*, *Conc. Magn. Reson.*, **1990**, 2, 197-212, specifically page 209.

which applies to a dynamical case of a spherical top undergoing random rotational motion[1]. Putting this formula first in terms of the quadrupolar coupling constant

$$R_1^Q = \frac{1}{T_1^Q} = \left[\frac{3\tau(2I+3)}{400I^2(2I-1)}\right]QCC^2\left[1+\frac{\eta^2}{3}\right]\left[\frac{2}{1+(\omega\tau)^2}+\frac{8}{1+(2\omega\tau)^2}\right]$$

into GAMMA nomenclature using the quadrupolar interaction constant yields

$$R_1^Q = \frac{1}{T_1^Q} = (2I+3)(2I-1)(\xi^Q)^2\frac{1}{20}\left[1+\frac{\eta^2}{3}\right]\left[\frac{1}{1+(\omega\tau)^2}+\frac{4}{1+(2\omega\tau)^2}\right] \quad (3\text{-}2)$$

This longitudinal relaxation equation predicts how the correlation time affects T_1 times based on the electric quadrupole. The following figure was generated by a GAMMA program[2] for various nuclei at with three differing quadrupolar coupling constants. Keep in mind that this simple treatment assumes that the system containing the spin moves as a spherical top with isotropic motions.

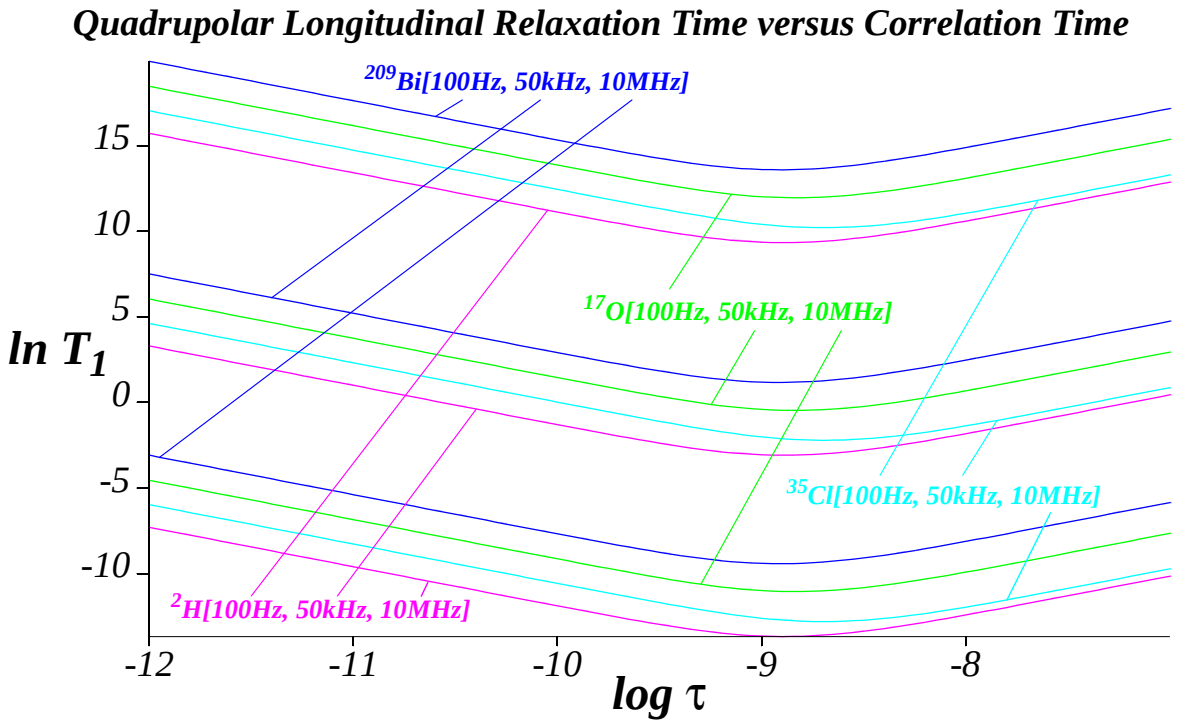


Figure 0-7 **Natural log of the quadrupolar longitudinal relaxation time versus the base 10 log of the correlation time. The field strength was 500 MHz & isotopes are noted with QCC values bracketed.**

Note that for small values of τ where $\omega\tau \ll 1$ (the extreme narrowing condition), R_1^Q increases and T_1^Q decreases linearly with the correlations time: as the molecule begins to slow, τ increases, the relaxation rate increases, and the relaxation time becomes shorter. The opposite is true when we

1. See "Calculation of Nuclear Spin Relaxation Times" by James L. Sudmeier, *et. al.*, *Conc. Magn. Reson.*, **1990**, 2, 197-212, specifically page 201, equation [25].
2. This figure was produced by the GAMMA program T1plot_QQ.cc listed at the end of this chapter.

are far away from the extreme narrowing, when $\omega\tau \gg 1$. Then, as the molecule further slows down (heading toward a solid) the electric quadrupole no longer provides a nice longitudinal relaxation pathway. It is then T_1^Q which increases linearly with τ. W e can estimate the spin lattice quadrupolar relaxation rate under extreme narrowing (EN) conditions. Recall that for extreme narrowing $\omega\tau \ll 1$. In these instances, equation (3-2) becomes

$$R_1^Q\big|_{EN} = \left[\frac{3\tau(2I+3)}{40I^2(2I-1)}\right]QCC^2\left[1+\frac{\eta^2}{3}\right] = (2I+3)(2I-1)(\xi^Q)^2\frac{1}{4}\left[1+\frac{\eta^2}{3}\right] \tag{3-3}$$

and it is apparent that the relaxation rate is proportional to the correlation time. Using the quadrupolar coupling constant previously calculated for a ^{2}H nucleus in toluene-d_3 assuming an axially symmetric quadrupole tensor and a correlation time of 1 picosecond we can directly calculate the spin lattice relaxation time expected in the extreme narrowing limit.

$$T_1^Q\big|_{EN} = \left[\frac{3\tau(5)}{40}\right]QCC^2 = \left[\frac{3}{8}(1\times10^{-12}\text{sec})4\pi^2(165\times10^3 Hz)\right]^{-1} = 2.481 \text{ sec}$$

Notice that this corresponds to $ln(T_1^Q) = 0.91$ for ^{2}H which can be roughly estimated from the previous plot.

4.6.3 Quadrupolar Transverse Relaxation

We now consider the transverse or T_2 relaxation expected from the electric quadrupole moment of a spin. For this simple treatment there exists an equation typically found in the literature[1].

$$R_2^Q = \frac{1}{T_2^Q} = \left[\frac{3\tau(2I+3)}{400I^2(2I-1)}\right]\left[\frac{e^2Qq}{h}\right]^2\left[1+\frac{\eta^2}{3}\right]\left[3+\frac{5}{1+(\omega\tau)^2}+\frac{2}{1+(2\omega\tau)^2}\right] \tag{3-4}$$

Again, this is restricted to the dynamical case of a spherical top undergoing random rotational motion. Putting in the quadrupolar coupling constant we have

$$R_2^Q = \frac{1}{T_2^Q} = \left[\frac{3\tau(2I+3)}{400I^2(2I-1)}\right]QCC^2\left[1+\frac{\eta^2}{3}\right]\left[3+\frac{5}{1+(\omega\tau)^2}+\frac{2}{1+(2\omega\tau)^2}\right]$$

and we now place the formula into full GAMMA nomenclature using the quadrupolar interaction constant.

$$R_2^Q = \frac{1}{T_2^Q} = (2I+3)(2I-1)(\xi^Q)^2\frac{1}{20}\left[1+\frac{\eta^2}{3}\right]\left[3+\frac{5}{1+(\omega\tau)^2}+\frac{2}{1+(2\omega\tau)^2}\right] \tag{3-5}$$

We now show graphically how T_2 varies with correlation time in accordance with equation (3-5). The figure below applies to a single spin system at 500 MHz[2].

1. Also found in the previous reference, “Calculation of Nuclear Spin Relaxation Times” by J.L. Sudmeier, S.E. Anderson, and J.S. Frye, *Conc. Magn. Reson.*, **1990**, 2, 197-212, this corresponds to page 201, equation [26].

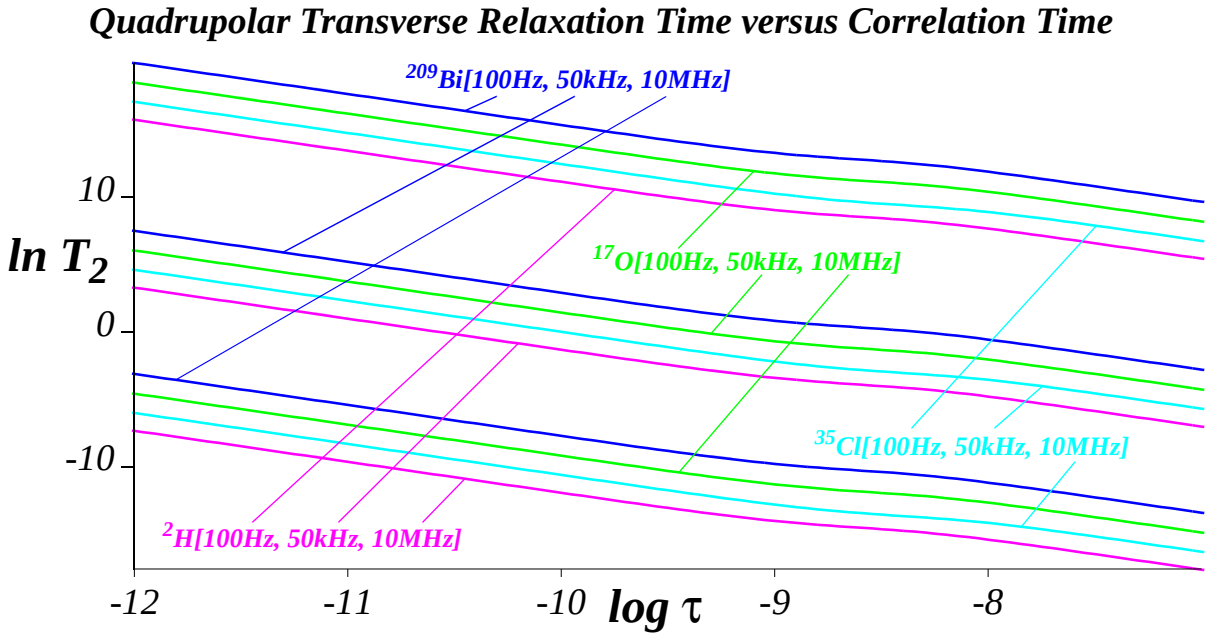


Figure 0-8 **Natural log of the quadrupolar transverse relaxation time versus the base 10 log of the correlation time. The applied field strength was set to 500 MHz. The isotopes are noted with the QCC values indicated in brackets.**

Under extreme narrowing, $\omega\tau \ll 1$, the transverse quadrupolar relaxation rate is

$$R_2^Q\big|_{EN} = \left[\frac{3\tau(2I+3)}{40I^2(2I-1)}\right]QCC^2\left[1+\frac{\eta^2}{3}\right] = (2I+3)(2I-1)(\xi^Q)^2\frac{1}{4}\left[1+\frac{\eta^2}{3}\right] \tag{3-6}$$

and comparison of this equation with that for $R_1^Q\big|_{EN}$ reveals that they are equivalent. Thus in the extreme narrowing limit, according to the inverses of equations (3-3) and (3-6)

$$\frac{T_1^Q\big|_{EN}}{T_2^Q\big|_{EN}} = \frac{R_2^Q\big|_{EN}}{R_1^Q\big|_{EN}} = 1 \tag{3-7}$$

4.6.4 Quadrupolar Relaxation Linewidths

The linewidths expected from quadrupolar relaxation may be estimated directly from the transverse relaxation times according to the following relationship.

$$LW_{hh}^Q = R_2^Q/\pi = 1/(\pi T_2^Q) \tag{3-8}$$

Here LW_{hh}^Q is used to indicate the quadrupolar related line-width at half-height.

2. This figure was produced by the GAMMA program listed at the end of this chapter, T2plot_QQ.cc, page 78.

4.6.5 Quadrupolar Relaxation Equations

We now group together the important equations regarding a the simple treatment of quadrupolar relaxation.

Quadrupolar Relaxation Equations

Interaction Constant

$$\xi_i^Q = \sqrt{\frac{6\pi}{5}}\frac{\delta_{zz}(i)}{2I_i(2I_i-1)} = \sqrt{\frac{6\pi}{5}}\frac{QCC_i}{2I_i(2I_i-1)} = \sqrt{\frac{6\pi}{5}}\frac{2\pi\nu_i^Q}{I_i(2I_i-1)}$$

Longitudinal Relaxation (Spin-Lattice)

$$R_1^Q = \frac{1}{T_1^Q} = (2I+3)(2I-1)(\xi^Q)^2\frac{1}{20}\left[1+\frac{\eta^2}{3}\right]\left[\frac{1}{1+(\omega\tau)^2}+\frac{4}{1+(2\omega\tau)^2}\right]$$

$$= \left[\frac{3\tau(2I+3)}{400I^2(2I-1)}\right]QCC^2\left[1+\frac{\eta^2}{3}\right]\left[\frac{2}{1+(\omega\tau)^2}+\frac{8}{1+(2\omega\tau)^2}\right]$$

Transverse Relaxation

$$R_2^Q = \frac{1}{T_2^Q} = (2I+3)(2I-1)(\xi^Q)^2\frac{1}{20}\left[1+\frac{\eta^2}{3}\right]\left[3+\frac{5}{1+(\omega\tau)^2}+\frac{2}{1+(2\omega\tau)^2}\right]$$

$$= \left[\frac{3\tau(2I+3)}{400I^2(2I-1)}\right]QCC^2\left[1+\frac{\eta^2}{3}\right]\left[3+\frac{5}{1+(\omega\tau)^2}+\frac{2}{1+(2\omega\tau)^2}\right]$$

Linewidth at Half-Height

$$LW_{hh}^Q = R_2^Q/\pi = 1/(\pi T_2^Q)$$

Extreme Narrowing

$$R_1^Q\big|_{EN} = R_2^Q\big|_{EN} = \left[\frac{3\tau(2I+3)}{40I^2(2I-1)}\right]QCC^2\left[1+\frac{\eta^2}{3}\right]$$

4.6.6 Quadrupolar Single Spin Relaxation

The following table[1] tabulates the quadrupolar relaxation parameters and linewidths expected at a

field strength of 500.12 MHz.

Table 6: Estimated Quadrupolar Relaxation Times @ 500 MHz

QCC (kHz)	tau (sec)	T_1 (sec)	T_2 (sec)	LW_{HH} (Hertz)	T_1 (sec)	T_2 (sec)	LW_{hh} (Hertz)
		I=1 (^{2}D, ^{14}N)			I=3/2 (^{35}Cl)		
1	10^{-12}	67547.5	67547.5	4.71239e-06	253303	253303	1.25664e-06
	10^{-11}	6755.28	6754.95	4.71225e-05	25331.1	25330.6	1.25662e-05
	10^{-10}	680.811	677.510	0.00047	2541.19	2536.14	.000126
	10^{-9}	117.128	83.4679	.003814	332.061	280.899	.001133
	10^{-8}	403.211	20.9295	.015209	638.710	71.7828	.004434
	10^{-7}	3928.61	2.24981	.141483	6004.09	8.42714	.037772
100	10^{-12}	6.75475	6.75475	.047124	25.3303	25.3303	.012566
	10^{-11}	.675528	.675495	.471225	2.53311	2.53306	.125662
	10^{-10}	.068081	.067751	4.69823	.254119	.253615	1.25509
	10^{-9}	.011713	.008347	38.1356	.033206	0.02809	11.3318
	10^{-8}	.040321	.002093	152.087	.063871	.007178	44.3434
	10^{-7}	.392861	.000225	1414.83	.600409	.000843	377.720
1000	10^{-12}	.067548	.067547	4.71239	.253303	.253303	1.25664
	10^{-11}	.006755	.006755	47.1225	.025331	.025331	12.5662
	10^{-10}	.000681	.000678	469.823	.002541	.002536	125.509
	10^{-9}	.000117	8.34679e-05	3813.56	.000332	.000281	1133.18
	10^{-8}	.000403	2.09295e-05	15208.7	.000639	7.17828e-05	4434.34
	10^{-7}	.003929	2.24981e-06	141483	.006004	8.42714e-06	37772.0
10,000	10^{-12}	.000675	000675	471.239	.002533	.002533	125.664
	10^{-11}	6.75528e-05	6.75495e-05	4712.25	.000253	.000253	1256.62
	10^{-10}	6.80811e-06	6.7751e-06	46982.3	2.54119e-05	2.53615e-05	12550.9
	10^{-9}	1.17128e-06	8.34679e-07	381356	3.32061e-06	2.80899e-06	113318
	10^{-8}	4.03211e-06	2.09295e-07	1.52087e+06	6.3871e-06	7.17828e-07	443434
	10^{-7}	3.92861e-05	2.24981e-08	1.41483e+07	6.00409e-05	8.42714e-08	3.7772e+06

1. This table was generated from the program T1T2_QQ.cc listed at the end of this Chapter. The output from the program was placed into this document as a table (in FrameMaker) by first placing the program output into a file then importing it as an ASCII file. The imported text is then converted into a Format B Table with the paragraphs treated as cells using 1 or more blank spaces as a cell. This new table is then unconverted (another Table option: convert to paragraphs) in a column by column fashion. This procedure allows a table ASCII output from the program as well as facile generation of the table in this text! It isn't as difficult as it sounds.

4.7 Quadrupolar Source Codes

T1plot_Q - Generate Plot of Quadrupolar T1 versus tau

```
/* T1plot_Q.cc ***************************************************************-*-c++-*-
**                                                                             **
**                  Example program for the GAMMA Library                      **
**                                                                             **
** This program constructs a plot of T1 versus tau for a single               **
** spin system under the effects of quadrupolar relaxation.                   **
** The calculations are performed using a simple analytic                     **
** formula for the T1 time which assumes that spin motion is                  **
** that of a spherical top under rotationally diffusive motion.               **
**                                                                             **
*********************************************************************************** */

#include <relax_Quad.h>

main (int argc, char* argv[])
 {
 cout << "\n\n\t\t\tGAMMA NMR Checking Program";
 cout << "\n\t\tQuadrupolar T1 Relaxation - Single Spin System\n\n";

 sys_dynamic dsys(1);                          // Set up a 1 spin system
 dsys.Omega(500.0);                            // Set Omega to 500 MHz.
 int npts = 101;                               // Use 101 points each T1 vs. tau
 String types[4];                              // Look at 4 isotope types
 types[0] = "2H";                              // Deuterium (I=1)
 types[2] = "35Cl";                            // Chlorine (I=3/2)
 types[1] = "17O";                             // Oxygen (I=5/2)
 types[3] = "209Bi";                           // Bismuth (I=7/2)
 double QCC[3];                                // Set up 3 QCC values
 QCC[0] = 100;                                 // 100 Hz
 QCC[1] = 50000;                               // 50 kHz
 QCC[2] = 10000000;                            // 10 MHz

 row_vector plot(npts), plots[16];             // Storage for plots
 double taui = 1.e-12;                         // Start at 10**-12 correlation
 double lntauinc = 5.0/double(npts-1);         // Increment tau 10**5 sec
 double lntau, tau, T1;

 int k=0;
 for(int iso=0; iso<4; iso++)                  // Loop through all isotopes
  {
  dsys.isotope(0, types[iso]);                 // Set spin isotope type
  for(int dz=0; dz<3; dz++)
   {
   dsys.Qdelz(0, QCC[dz]);                     // Set system QCC
   tau = 1.0e-12;                              // Start at 1 psec tau
   lntau = -12.0;
   for(int i=0; i<npts; i++)                   // Loop through all points
    {
    dsys.taux(tau);                            // Set tau value of system
    T1 = T1_QQ(dsys, 0);                       // Calculate new T1 value
    if(i == 0)                                 // Output initial & final values
     {                                         // to screen so plots discernable
     cout << "\n" << types[iso] << " - " << dsys.Qdelz(0);
     cout << ": initial ln(T1) = " << log(T1);
     }
    else if(i == npts-1)
     cout << ", final ln(T1) = " << log(T1);
    plot.put(log(T1), i);                      // Store new T1 value
    lntau += lntauinc;                         // Increment log of tau
    tau = pow(10.0, lntau);                    // Determine next tau
    }
   plots[k] = plot;                            // Store this T1 vs. tau plot
   k++;
   }
  }
FM_1Dm("T1Qplot.mif",k,plots,19,14,-12,-7);    // Output all plots to FrameMaker
cout << "\n\n";                                // Keep screen nice
}
```

T2plot_Q - Generate Plot of Quadrupolar T1 versus tau

```
/** T1plot_Q.cc ************************************************-*-c++-          *-
**                                                                              **
**
Example program for the GAMMA Library                                          **
**                                                                              **
** This program constructs a plot of T1 versus tau for a single                 **
** spin system under the effects of quadrupolar relaxation.                     **
** The calculations are performed using a simple analytic                       **
** formula for the T1 time which assumes that spin motion is
**
** that of a spherical top under rotationally diffusive motion.                 **
**                                                                              **
*******************************************************************************/

#include <relax_Quad.h>

main (int argc, char* argv[])
 {
 cout << "\n\n\t\t\t GAMMA NMR Checking Program";
 cout << "\n\t\tQuadrupolar T2 Relaxation - Single Spin System\n\n";

 sys_dynamic dsys(1);                          // Set up a 1 spin system
 dsys.Omega(500.0);                            // Set Omega to 500 MHz.
 int npts = 101;                               // Use 101 points each T2 vs. tau
 String types[4];                              // Look at 4 isotope types
 types[0] = "2H";                              // Deuterium (I=1)
 types[2] = "35Cl";                            // Chlorine (I=3/2)
 types[1] = "17O";                             // Oxygen (I=5/2)
 types[3] = "209Bi";                           // Bismuth (I=7/2)
 double QCC[3];                                // Set up 3 QCC values
 QCC[0] = 100;                                 // 100 Hz
 QCC[1] = 50000;                               // 50 kHz
 QCC[2] = 10000000;                            // 10 MHz

 row_vector plot(npts), plots[16];             // Storage for plots
 double taui = 1.e-12;                         // Start at 10**-12 correlation
 double lntauinc = 5.0/double(npts-1);         // Increment tau 10**5 sec
 double lntau, tau, T2;

 int k=0;
 for(int iso=0; iso<4; iso++)                  // Loop through all isotopes
  {
  dsys.isotope(0, types[iso]);                 // Set spin isotope type
  for(int dz=0; dz<3; dz++)
    {
    dsys.Qdelz(0, QCC[dz]);                    // Set system QCC
    tau = 1.0e-12;                             // Start at 1 psec tau
    lntau = -12.0;
    for(int i=0; i<npts; i++)                  // Loop through all points
      {
      dsys.taux(tau);                          // Set tau value of system
      T2 = T2_QQ(dsys, 0);                     // Calculate new T2 value
      if(i == 0)                               // Output initial & final values
        {                                      // to screen so plots discernable
        cout << "\n" << types[iso] << " - " << dsys.Qdelz(0);
        cout << ": initial ln(T2) = " << log(T2);
        }
      else if(i == npts-1)
        cout << ", final ln(T2) = " << log(T2);
      plot.put(log(T2), i);                    // Store new T2 value
      lntau += lntauinc;                       // Increment log of tau
      tau = pow(10.0, lntau);                  // Determine next tau
      }
    plots[k] = plot;                           // Store this T2 vs. tau plot
    k++;
    }
  }
FM_1Dm("T2Qplot.mif",k,plots,19,1,-12,-7);     // Output all plots to FrameMaker
cout << "\n\n";                                // Keep screen nice
```

T1T2_Q - Generate Table of Quadrupolar Relaxation Values

```
/* T1T2_Q* T1plot_Q.cc ************************************************-*-c++-  *-
**                                                                          **
**
Example program for the GAMMA Library                                       **
**                                                                          **
** This program constructs a plot of T1 versus tau for a single             **
** spin system under the effects of quadrupolar relaxation.                 **
** The calculations are performed using a simple analytic                   **
** formula for the T1 time which assumes that spin motion is
**
** that of a spherical top under rotationally diffusive motion.             **
**                                                                          **
******************************************************************************  */

#include <relax_Quad.h>

main (int argc, char* argv[])
 {
 cout << "\n\n\t\t\tGAMMA NMR Checking Program";
 cout << "\n\t\tQuadrupolar Relaxation - Single Spin System\n\n";

 sys_dynamic dsys(1);                          // Set up a 1 spin system
 double bigO;                                  // Set the spectrometer frequency
 query_parameter(argc, argv, 1,
   "Spectrometer Frequency (MHz)? ", bigO);
 dsys.Omega(bigO);
 String iso;                                   // Set the isotope type
 query_parameter(argc, argv, 2,
          "Spin Isotope Type? ", iso);
 dsys.isotope(0, iso);
 double QCC[4];                                // Set up 4 QCC values
 QCC[0] = 1.0e3;                               // 1 kHz
 QCC[1] = 100.0e3;                             // 100 kHz
 QCC[2] = 1.0e6;                               // 1 MHz
 QCC[3] = 10.0e6;                              // 10 MHz
 double tau;                                   // Variable for tau
 double taui = 1.e-12;                         // Start at 10**-12 correlation
 double R1,R2,T1,T2,LW;
 cout << "\n\n\tQuadrupolar Relaxation of "
     << dsys.symbol(0) << "\n";
 cout << "\nQCC\ttau"
     << "\tT1\tT2\tLWHH";
 for(int i=0; i<4; i++)                        // Loop over the 4 QCC values
   {
   dsys.Qdelz(0, QCC[i]);                      // Set system delzz
   dsys.taux(taui);                            // Set initial correlation time
   tau = dsys.taux();
   for(int j=-12; j<-6; j++)                   // Loop over different taus
     {
     R1 = R1_QQ(dsys, 0);                      // Calculate R1
     R2 = R2_QQ(dsys, 0);                      // Calculate R2
     T1 = 1.0/R1;                              // Calculate T1 from R1
     T2 = 1.0/R2;                              // Calculate T2 from R2
     LW = R2/PI;                               // Calculate Linewidth from R2
     cout << "\n" << QCC[i]                    // Output in form which can be
          << "\t" << tau                       // placed into a FrameMaker table
          << "\t" << T1
          << "\t" << T2
          << "\t" << LW;
     tau *= 10.0;                              // Increase tau by 10x
     dsys.taux(tau);
     }
   }
 cout << "\n\n";                               // Tidy up output
 }
```